THE BEST SELLING COMPUTER PROJECTS MAGAZINE FEBRUARY 1985

# ELECTRONICS & COMPUTING

MONTHLY AN EMAP PUBLICATION

15 3a REF 34 39 55 ROBOT 24

USA $2.95 Germany D6.00 Singapore S$4.95 95p

PLUS Slot car timer project

## THE GREAT MICRO RACE

### Who's got the winning formula ?

33 NEW ADDRESS FOR MAGAZINE

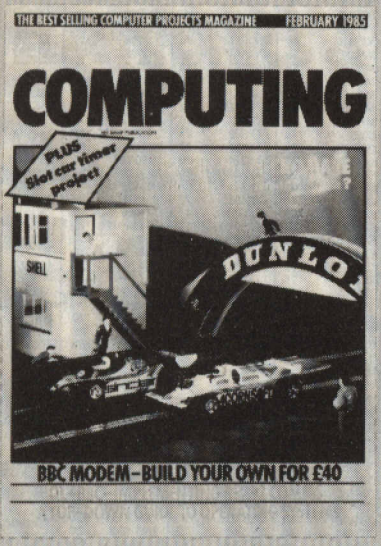

## BBC MODEM – BUILD YOUR OWN FOR £40

## QL+BBC – IMPLEMENTING RS232 COMMS

## A TOP-DOWN GUIDE TO OPERATING SYSTEMS

# ELECTRONICS & COMPUTING Contents

Vol. 5 Issue 2




COVER PHOTO BY ROB BRIMSON

Electronics & Computing Monthly
Priory Court, 30-32 Farringdon Lane,
London, EC1R 3AU

**Editorial 01-251-6222**
**Editor** Gary Evans
**Deputy Editor** William Owen
**Production Editor** Liz Gregory
**Administration** Serena Hadley

**Advertising 01-251-6222**
**Advertisement Manager** Anthony Herman
**Chief Executive** Richard Jansz
**Classified** Tracey Keighley
**Advertising Production** Yvonne Moyser

**Production 01-251-6222**
**Art Editor** Jeremy Webb
**Make-up** Time Graphics
**Publisher** Alfred Rolington

**Distribution**
EMAP National Publications

**Published by**
EMAP Business and Computer Publications

**Printed by**
Riverside Press, England

**Subscriptions**
Electronics & Computing Monthly, (Subscription Department), Competition House, Farndon Road, Market Harborough, Leicestershire.

Electronics & Computing Monthly is normally published on the 13th day of each month.



## PROJECTS



## FEATURES



## REVIEWS



## PLUS



**And within the pages of *Your Robot***
**The latest robotics news plus how to build a low cost LEGO arm.**

**STOP PRESS . . .** for the price of the Heart Rate Monitor kit see Cirkit's advert elsewhere in this issue.

# A TAX ON FREEDOM

A change of style this month as we comment not on news from the computer industry but on the effects of the Chancellor's 'threat' to extend the scope of Value Added Tax to include the printed word. The taxing of publications is an established practice in many EEC countries and one justification for the introduction of VAT in the UK is likely to be that it will bring us into line with the rest of Europe. Freedom of the press, both in terms of the right of free expression and in freedom from taxation is though, a long held tradition in this country. As adverts prepared by the PPA (Periodical Publishers Association) will tell you, the last tax on the press was abolished in 1855, and the freedom from taxation has been jealously guarded for 130 years.

The tax imposed on magazines throughout the EEC varies considerably, and is usually fairly low. There are however strong reasons to believe that in this country the Chancellor would opt for the simple approach and impose the standard 15% tax on all printed media. The effects of the tax if imposed will affect the various sections of the publishing world in different ways but all areas will suffer.

In the case of *E&CM* the immediate effect would be to add almost 15p to the cover price. While we would hope this would not cause us to lose too many of our readers, it could make the magazine too costly for some to buy on a regular basis. Any resulting loss in circulation would affect the quality of service we would be able to offer in the future.

The effects of the threatened tax go further than simply increasing the cover price of magazines however. In addition we would have to charge our advertisers an additional 15% due to VAT. Larger VAT registered businesses would be able to claim the tax back from the Government but, like so many specialist magazines, a large number of *E&CM's* advertisers are small concerns not registered for the purposes of VAT. These people would face a 15% increase in the rates charged to them and in the face of this could well decide that they could not afford to advertise. This loss of revenue would also begin to affect the service we were able to offer our readers.

The imposition of the tax would thus be almost certain to have an adverse effect on *E&CM* as it will all areas of the publishing world. The threatened tax has been called a tax on knowledge, it could well put some publications out of business and will so limit the choice of material available to the public – a tax on freedom.

We would ask that if you have the chance you back the view that VAT should not be applied to the printed word – if you can, write to your MP and let them know that you feel there should be no tax on reading. We hope that the Government will be persuaded not to turn the clock back to the 19th century and will maintain the principle that in the UK there will never be a tax on knowledge.

**GARY EVANS**


# CHARTER FOR CHEATS

Are you a cheat? Clumsy? A birdbrain? If the answer is yes then Slomo will be the best thing since the marked card; it enables you to cheat at computer games.

Slomo is a device which slots into the expansion port of the Spectrum, Electron, BBC micro and Commodore 64. It allows all Spectrum games and '95%' of BBC and Commodore games to be played at an infinite variety of speeds – and therefore skill levels. In addition to the speed control Slomo has freeze frame and slow motion switches.

The manufacturers point to the benefits Slomo could bring to the disabled or to young children in education. There are also applications which *E&CM* readers may find useful: in software development – animated graphics can be viewed pixel by pixel; slowing down the scroll of program listings; obtaining photographic copies of moving screen displays; freezing a program if any interruption occurs.

Slomo is available by mail order at £14.95 from Cambridge Computing Research, 61 Ditton Walk, Cambridge CB5 8QD.

# Optimum QL storage

STOP, from Digitex Computers, is one of the drips in the dribble of third party supplies of QL software now coming onto the market.

STOP (STorage Optimiser) uses data compression techniques to reduce Microdrive files to about half the normal size, which for users writing programs of more than about 60K should be a valuable utility. STOP will also reduce the price of storage on the expensive QL and Spectrum Microdrive cartridges.

An upgraded version is soon to be released for use on Winchester hard disc systems; this will enable complete back-ups to be made from Microdrives or floppy discs. Digitex Computers, 1 Amwell House, The Woodlands, Isleworth, Middx TW7 6NX.

# Osborne snips

Such a pity that it takes a market crash to reduce the price of a computer to an affordable level. Future Management, the UK distributor of Osborne computers, is offering the – legendary, as they call it – Osborne 1 for £499 plus VAT.

The standard version of this portable computer runs CP/M 2.2; Wordstar, Mailmerge, MBasic, CBasic and Supercalc are included in the price. The system has twin double-density 185K disc drives, but the screen display is of only 52 columns. For the full 80 column display you will have to fork out another £160. Future Management, 0908 615274.

# Spectrum spectrum analysis


AWR technology have addressed themselves to the problem of the high cost of digital storage oscilloscopes and come up with a device which uses the Spectrum's processor instead of its own.

The Microview – which also acts as a spectrum analyser – is said to offer a wide range of features found on more expensive devices. It uses machine code routines for fast plotting of data and has comprehensive menu options for analysis of waveforms.

The 'scope, which is priced at £140.00, is aimed at the amateur electronics enthusiast and the educational market. Versions for the BBC micro and Apple computers are currently at the design stage. AWR Technology 0227 459000.

# CHRIS CURRY WINS E&CM COMPETITION

You may recall that in our December 1984 issue we asked readers to speculate on the specification of the new computer Acorn were widely reported to be developing as a replacement for the BBC micro. Just before the closing date of our competition a late entry arrived in the form of a statement from joint Acorn MD, Chris Curry, which rather spoilt the spirit of the endeavour.

Curry disclosed to the Financial Times that the new computer will mean that Acorn will have hardware that should, once again, bring the company into competition with Sir Clive and his QL both in its Sinclair and ICL OPD incarnations. The new machine will be designated the C, the C standing for communications. It will be based on a 16-bit pin compatible version of the 6502 processor which is called the 65C816. This device has a 24 bit address bus and can thus directly address up to 16 MB of memory. Two modes of operation mean that the MPU can either function as a 6502 emulator or as a full 16-bit processor.

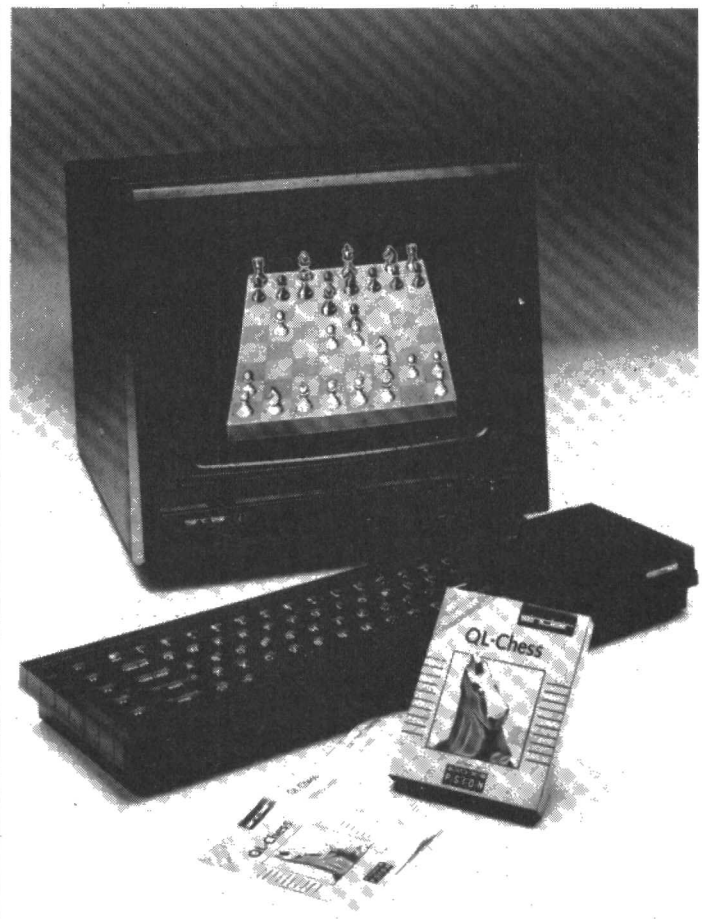
The former feature means that the C will be able to support BBC BASIC and thus the machine will offer compatability with both the BBC B and the Electron. The computer is not to be marketed under the BBC name but, as with the Electron, the company are likely to strive to gain maximum mileage from their association with the Beeb.

At present it is expected that the computer will feature a built-in telephone handset, as the OPD, and at least in some variants, View and Viewsheet software built in 3.5in drives and a flat screen display.

Present predictions are that basic versions of the C will retail at between £500 and £800 and that first production machines will be released in the Spring although, as with any project of this nature, it is almost inevitable that various delays will push the release back until the Summer.

A machine of the C's specification and price will once again mean that Acorn have a product that is likely to succeed in the evermore cut-throat computer market both in the home and business fields.

Perhaps, as a final thought, we should run a sweepstake as to how many publishers will be rushing to produce the Your C, C User et al magazines.




# IT'S YOUR MOVE MATE

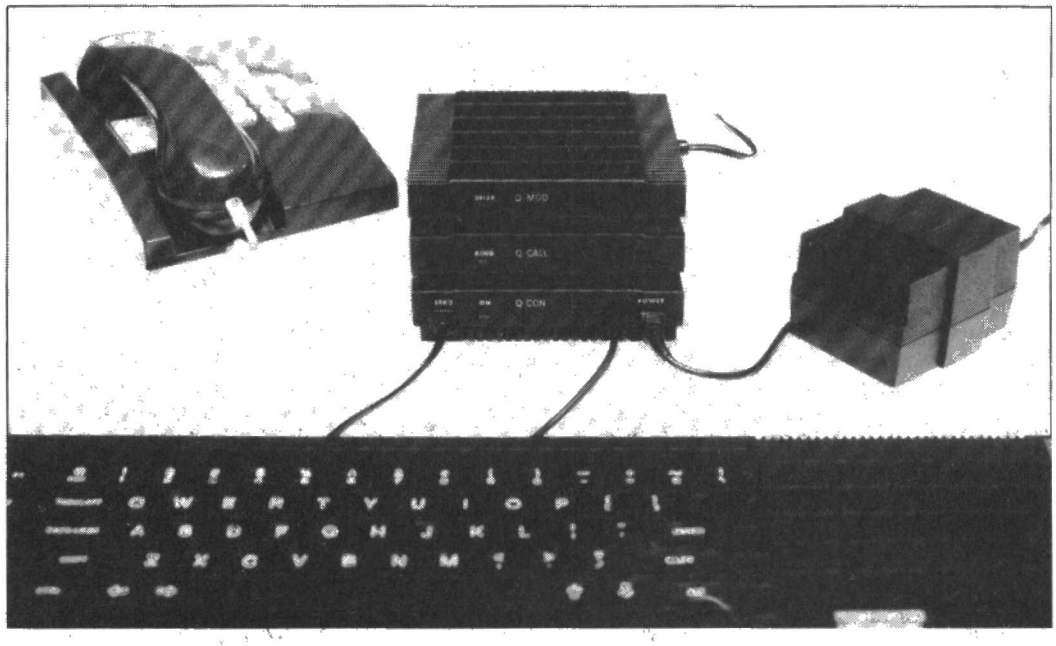
**The three dimensional QL chess game from Psion is now available in its final form. We've seen the production version and it lives up to most of the Sinclair hyperbole: 'an outstandingly powerful program with quite remarkable screen graphics'! The program offers an array of analysis commands, an opening book of 4000 moves, eight levels of problem solving, but not the 28 levels of play claimed by Sinclair: there only appear to be ten. The player can also ask the computer for the odd hint, or take back moves. Look out for a review in future issues.**

# OPD look out, QL + QCOM arrives

Software products may be few and far between, but the proliferation of Q peripherals continues unabated. QCOM (consisting of QCON, QMOD and QCALL) is a no-holds-barred communications package from OEL which turns the QL into an intelligent desk top terminal.

QCON is a single chip microcomputer which controls the flow of data, emulates a DEC VT100, and provides an RS232/V24 serial interface for connection to an asynchronous (transmits signals at irregular intervals) modem at speeds of between 75 and 9600 baud. QCON also includes software to control each of the three components of the QCOM system.

QMOD is a modem, with a V23 serial interface for direct connection to the BT system. Communication is at 1200/75 baud or 1200/1200 baud half-duplex. The third component is QCALL, which adds auto-dial and auto-answer facilities to the modem. The complete system is priced at £210. OEL Limited 0768 66748.



# U.S. REPORT



**In a new bi-monthly column, Marshal Rosenthal reports on the activities of the various players in the American micro market place.**

Greetings from America. This column begins a bi-monthly look at the US computing world, a small world that affects us all.

## Robots

Computerized robots have always fired the imagination, and who hasn't wanted to have the power of commanding a machine that walked and talked. But until recently, the prices for such sophisticated machines were prohibitive for the average consumer. So he/she made do with toys that bumped off walls.

The Androbot corporation of San Jose, California has changed all that. Known for a series of highly intelligent (and expensive) robots, they have now introduced their Friendly Robot Educational Device, or FRED for short. Fashionably attired in black and white, with a geometric shape like his older brother TOPO, FRED costs $499.00. The latest in computerized robots, he stands only one foot tall. He can be directed by remote control from a hand-held device. He'll also respond to an Apple or Commodore 64 computer using an optional interface with FREDSOFT, a Logo-language program which sells separately for $79.00. With the computer's help, FRED can draw shapes and letters as well as move about. His tether is a 16 foot infra-red light link, but he can break free of this restriction because his memory buffer is large enough to hold a number of commands.

FRED is smart too. His digitized human voice knows 57 words, and he'll tell you he's "tired" if his batteries are running down. Forget about remote-controlled toys, this is the real thing.

## Controlling piracy

One of the big issues that keeps popping in and out is software protection. Old day Pirates waved cutlasses and took your pocketbook, today they hunch over keyboards as sectors are stripped, and programs torn apart. The lack of conformity in the US (and overseas) industry has kept any kind of uniform practice from being adopted. System users haven't been satisfied with such schemes as "hard fingerprints" (a tiny hole cut in the disk by a laser), or special modules that go into joystick or control ports.

The Vault corporation of California produces Prolok, a copy protection technique that is added to third party software. Vault is now marketing a booby-trap version called Killer Prolok. Killer is reported to create a variety of nasty effects if the disk it protects is copied. Chairman W. Kraig Brotby, won't elaborate, but he did indicate that some of the results of Killer in action could be "planting a worm in the user's operating system, causing the computer to crash or malfunction." Killer might also reformat a hard disk or cause other strange effects, and Brothby says that such disks may not be identified.

Another company, Defendisk of Denver, says that their protection scheme booby-trap might make random changes in the operating system or even copy garbage to other programs. Better/worse yet – the booby-trap may wait a while before wreaking havoc. Now that's a real fear threat.

The problems that might occur are lawsuits, a result of a user having his system destroyed because of ignorance, not as a result of intended piracy. Still there will always be someone who can crack any program, so the best answer may be that forwarded by the Association of Data Processing Service Organizations. They advocate hardware protection. Their system consists of a small piece of hardware connected to the computer by an RS-232 serial port. Shown last October, the "Lock" device is a small box with slots that takes a key which, by interacting with integrated circuits, allows the software designed around it to run on that particular system. Cost of the "lock" is estimated at $35-50, with the key at $3. This seems to be a reasonable solution, but the computer world rarely operates on logic alone. Only time will tell.

## The end of Telex?

The 50 year old telex system may go to the chopping block soon. Western Union (which has quite a bit invested in telex systems) has made a major commitment to EasyLink, the new electronic mail service. It lets customers use a computer or terminal to send and receive electronic mail and telexes through dial-up services. Started in 1982, EasyLink has about 100,000 subscribers – making it the largest in the U.S. EasyLink transmits over the existing Western Union lines, and plans are to add 70 million dollars to the system.

Western Union will be delivering documents internationally through DHL Worldwide courier express in 1985. The projected domestic delivery time will be 2 hours, and internationally, next day. The UK has agreed to let Western Union operate EasyLink through a partially owned affiliate, and it will be fully integrated with the US department.

Elsewhere, telex will still endure due to poor phone lines which can't handle high speed transmissions. But among the western world, the end of telex seems to be on the horizon.


**Since this column was written, Acorn have cut back on their US activity.**

## Acorn advertising hits big time

Acorn computers are making their second year stab into the lucrative US market. The company is concentrating mostly on eduction, targeting towards the elementary and secondary schools.

Acorn has about 4% of this market, due partly to the quality of the machine, and mostly to the respect American educators have for the British school system. New ads are focusing on the credibility and track record in the UK, as well as the machines networking ability. Acorn feels that, with 85% of the UK market under their belt, their computer will make real inroads in the States.

So that's about it for now. If you have any areas of interest that you'd like to see explored here, please feel free to write to me. Send your correspondence to Marshal Rosenthal, c/o Electronics & Computing Monthly, Priory Court, 30-32 Farringdon Lane, London EC1R 3AU.

# BREAKING THE PRICE BARRIER –

## THE £40 MODEM

### Andy Green's home brew design for a BBC micro compatible modem that feature both auto-dial and auto-answer options.

While the price of modems has shown a healthy downward trend for some time now, they are still rather expensive items of equipment. Prices range from the (just) sub £100 level to, well you've probably all seen some models with price tags in excess of £250. In addition, the majority of readily available modems are relatively unsophisticated in that they make no provision for an auto-dial facility and, of less note, an auto-answer option.

Our modem manages to offer a full specification modem complete with auto-dial and answer capabilities for the amazingly low price of only £40. We have managed to achieve this very low price by basing the design on a second-hand, yet full spec, ex-Telecom modem. Since these devices were in fact made by Telecom, in theory there should not be any problem with approval should there?

## Cabling up

Most people will only ever want to use the Display Electronics 2B as a straight modem with auto-dial. The auto-answer bit is only really useful for those wishing to set up their own Bulletin Board.

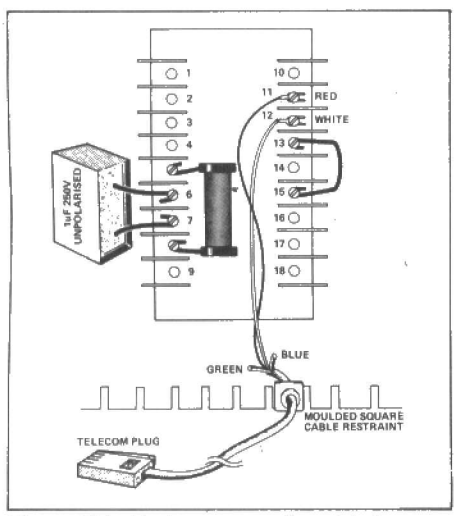



*Figure 2(c). Modem connection details – see text.*

If AA is not required then connection of the modem to the Beeb just entails making up a special lead and typing in the software. For those needing AA it is only necessary to wire up an extra two components to the back of the modem.

Taking the cable first, then. This consists of a 25-way D plug for the modem end, and a 15-way D and 5 pin DIN domino plug for the Beeb end. The domino plugs into the BBC's RS423 port, and the 15-way D into the Joystick port. Inside the 15-way D are six components. Two 5V1 Zeners, two 1N4148 diodes and two 1k2 resistors, as in **Figure 1** (see page 18).

When you're making the cable, do remember to put the DIN plug cover on the wires before soldering it together! While we're on the subject, when it comes to plugging the DIN plug into the RS423 port, you may notice that there are actually two possible ways of inserting the plug. The trick is that the notch on the metal part of the plug should be on the "Joystick port" side of the computer, and not with it pointing 180° away.

Assuming you've made the computer lead, let's get on with the telephone side of things. Before doing anything, make sure the modem is unplugged from the mains. Also check that your Telecom plug to spade lead isn't plugged into the 'phone system yet.

Unscrew the four screws at the back of the modem as in **Figure 2a.** The rear panel will come off, revealing a large screw-terminal board. This is marked clearly on **Figure 2b.**

You'll see that the screw terminals are marked with a number from 1 to 18. The blown-up picture in **Figure 2c** shows the Auto-answer components as well as the 'phone line connection.

Now fit the red and white wires from the Telecom plug on to terminals 11 and 12, and push the moulded square into one of the "fingers" to clamp the cable.

Use snips to crop the blue and green leads very near the moulded square. These are not connected to anything in a normal 'phone system.

The modem will now appear to the 'phone system as just another telephone. The Telecom plug at the end of the cable you've just connected up is just like a telephone; it'll plug into any Telecom standard 'phone socket.

If you don't want Auto Answer facilities, that's it. Replace the back panel and jump to the section marked "Testing".

If you do need them, then fit a small length of wire between terminals 13 and 15. Before you fit the capacitor, check it. It must be a 1uF unpolarised capacitor, rated at at least 250V, preferably 350V or higher, although 250V is perfectly adequate. Unpolarised means non-electrolytic . . . which means it doesn't care which way round it goes; bipolar means the same.

If it's working correctly, screw it between terminals 6 and 7, as in **Figure 2c.**

Finally, screw the thermister between terminals 5 and 8.

Now check to see that all the wires are well out of each other's way, and that the thermister and capacitor are pushed back slightly so as not to touch the back panel when it is replaced.

Check everything against **Figure 2c,** the replace the back panel.

All that remains now is to type in the software!

## Testing

Check the wiring of your cable again, especially the direction of the diodes in the 15-way D. Then plug in the two D plugs and the RS423 DIN plug. Don't connect the modem up to the telephone socket yet. If your modem has not already got a mains plug, fit one on and, yes, plug it in!

This should be an anticlimax – nothing should happen! It is possible that you'll hear a very quiet hum from the transformer inside the modem – and maybe when you switched on there was a couple of abortive clicking noises. This is nothing to worry about.

Now unscrew the two screws at the bottom of the front panel and lift it up. Place the manual selector on the left to "AUTO" and the one on the right to "DATA", that is if they are not there already. Also turn the "CARR FAIL LEVEL" potentiometer fully clockwise.

Now, on the computer type *FX2,2 <return>. As you press return, there should be a soft click from somewhere deep in the modem. If you type *FX2,0 it will click again.

This is the "line seize" relay going in and out, under computer control. *FX2,2 will bring it in and seize the line, while after a BREAK or after a *FX2,0 the relay is off and the line clear. This "seizing of the line" is exactly the same as picking and hanging up the receiver on a normal telephone. Note that just as on a normal telephone, if the receiver is off-hook (ie line seized) no-one can ring you up – they'll get "en-

**TABLE 1. BBC/2B control and data protocols.**

| D25 pin | Line from Modem | To BBC | READING | WRITING |
|---|---|---|---|---|
| 2 | TRANSMIT DATA | DATA OUT (RS423) | — | Write to as serial printer |
| 3 | RECEIVE DATA | DATA IN (RS423) | RS423 in at Buffer | — |
| 8 | CARRIER DETECT | PB4 (internal VIA) | (?&FE40)AND 16<br>=16=Present<br>= 0=Not present | — |
| 11 | SELECT TRANSMIT FREQ | CB2 (internal VIA) | — | ?&FE4C=&C4=Call tones<br>?EFE4C=&E4=Answer tones |
| 20 | DATA TERMINAL READY | RTS (RS423) | — | *FX2,2=connect Modem to line<br>*FX2,0=disconnect |
| 22 | RING INDICATOR | PBS (internal VIA) | (?&FE40)AND 32<br>=32=Ringing<br>= 0=Not ringing | — |

gaged".

Now key in *FX2,0 to make sure the modem is "off-line", and execute the terminal program in **Listing 1** but still do not plug the modem into the 'phone system yet.

You should get a nice little menu to come up, listing the first 17 numbers in the data statement. The blue bar can be moved up and down with the equivalent cursor keys. Select a number with the blue bar, and press the "copy" key. If you have more than 17 numbers in the data statements, use the left/right cursor keys to move back/forward a page of 17.

After a short pause, the modem should start clicking away at a heavy rate of knots. The pattern shouldn't be entirely unfamiliar – the relay is emulating the function of a normal telephone's rotary dial mechanism.

After it's "dialled" all the digits, the bottom line on the screen will say "awaiting carrier", which means it's waiting for an answer from the modem you just "called". This time, it'll just sit there forever, since we haven't connected the modem to the 'phone lines yet.

If all seems to be well then press ESCAPE on the BBC to return to the menu, which ensures that the modem is not "off hook", then plug the Telecom plug from the modem into your Telecom 'phone socket. Again, nothing much should happen.

Now select and dial DISTEL, the 5-line database run by Display Electronics. Any of the databases in the list will do, but DISTEL is the least likely to be engaged.

Everything will happen as before, and the "awaiting carrier" message will appear. But what exactly is the "carrier"?

The carrier is a tone from the answering modem. Inside the 2B there's a filter circuit to detect this particular tone and relay it's presence to the computer.

However, it is not the best filter in the world, and has a habit of registering the carrier as present when listening to dial tones etc. We can, though, turn this to our advantage. While the computer is awaiting the carrier from the answering modem it plays the current status of the "carrier detect" line through the computer's loudspeaker, so you can hear the dial tones etc.

This will not work very well for distant lines, since the small dial tone signals are not powerful enough to falsely trigger the carrier filter, but it is fine for most lines. The computer will only register the distant modem as present after around two seconds of unbroken carrier.

This means that it will ignore both the ringing and the engaged tones since these have a break in them as part of their pattern well within two seconds.

**LISTING 1. Terminal program.**

```
    8 QW%=0:MODE7:DIM RR% 256
   80 CR$=CHR$13+CHR$10:name$="ANDY"+CR$
+STRING$(20,CHR$0)+"GREEN"+CR$:*FX225,12
8
   90 temp=&70:temp1=&71:temp2=&72:start
=&73:DIM E% 400:PROCasm
   95 ONERROR IFERR=17THENMODE7:GOTO100
ELSE REPORT:PRINT;" at line ";ERL:END
  100 PX=0:RESTORE:*FX2,0
  110 CLS:PRINTCHR$132;CHR$157;CHR$135;C
HR$141;TAB(13);"2B Autodialler";CHR$132;
CHR$157;CHR$135;CHR$141;TAB(13);"2B Auto
dialler"
  120 FORI%=1TO17:K%=I%:READA$,B$:IFA$="
*"THENI%=18:K%=K%-1  ELSE PRINT"  "+CHR$
129;I%;17+P%;TAB(5);CHR$131;A$;TAB(24);C
HR$130;B$;CHR$131
  130   NEXT:PRINTTAB(0,23);CHR$129;CHR$
157;CHR$131;"Cursor Keys to select, COPY
 to dial";:Q%=1:PRINTTAB(0,4);CHR$132;CH
R$157;CHR$134;:*FX4,1
  140 A%=GET-135:IFA%<0THEN140 ELSE IFA%
=0THEN200
  150 IF(A%=3)AND(Q%<K%)THENPRINTTAB(0,3
+Q%);"  ";CHR$129;TAB(0,4+Q%);CHR$132;CH
R$157;CHR$134;:Q%=Q%+1:GOTO140
  155 IF(A%=4)AND(Q%>1)THENPRINTTAB(0,3+
Q%);"  "CHR$129;TAB(0,2+Q%);CHR$132;CHR$
157;CHR$134;:Q%=Q%-1:GOTO140
  160 IFNOT(A%=1)AND(P%>0)THEN170 ELSE
 P%=P%-1:RESTORE:IFP%=0THEN110 ELSE FORT
%=1TO17*P%:READA$,B$:NEXT:GOTO110
  170 IF(A%=2)AND(A$<>"*")THENP%=P%+1:GO
TO110
  180 GOTO140
  200 PRINTTAB(3,23);STRING$(36," ");:X%
=25:Y%=0:REPEAT:A%=?(&7C00+(40*(Q%+3))+X
%):Y%?buf=A%:X%=X%+1:Y%=Y%+1:UNTILA%>128
:?&start=(40-Y%)/2:CALLdial:IF?temp2 THEN
VDU7:GOTO100
  202 *FX7,3
  203 *FX8,3
  204 *FX5,2
  210 PRINTTAB(3,23);STRING$(36," ");TAB
(12,23);"AWAITING CARRIER":VDU2:CALLtran
s:VDU3:IF?temp =0THEN100
  500 SOUND&11,0,0,1:U%=0
  515 *FX4,0
  520 *FX15,0
  530 MODE0:VDU19,0,4,0,0,0:A%=13B:X%=2:
Y%=17:CALL&FFF4
  540 IF(ADVAL(-1)<=0)OR(ADVAL(-3)<=0)TH
EN545
  542 Y%=GET:IFY%<128THENCALL&FFF4 ELSE
GOTO600
  545 IF(?&FE40 AND16)THEN549
  546 IFU%=0THENU%=1:GOTO550 ELSE *FX2,0
  547 COLOUR129:COLOUR0:PRINT "Carrier
Dropped.";:COLOUR1:COLOUR128:PRINT" PRE
SS ANY KEY...":U%=GET:MODE7:GOTO100
  549 U%=0
  550 *FX2,1
  560 IFADVAL(-2)=0THEN570 ELSE Z%=GET:I
FZ%>3ITHENVDU1% ELSE IF(Z% AND(Z%-14)T
HENVDUZ%
  570 *FX2,2
  580 GOTO540
  600 COLOUR129:COLOUR0:IFY%<128THENPROC
chuk(name$,0):GOTO699
  610 IFY%<>129THEN660 ELSE INPUT "Type
 Filename: " F$:S%=580+F$+CHR$13:X%=&00:Y
%=&5:A%=&40:R%=USR(&FFCE)AND255:IFR%=0TH
ENPRINT"Not Openable":GOTO699
  620 S%=EXT#R%:PRINT ;S%;" Bytes long":
C%=0:FORT%=1TOS%:A%=BGET#R%:IF(A%<>13)TH
EN630
  622   IF(C%=1)AND(T%=S%)THENPROCchuk(C
R$,-1):GOTO650 ELSE IF(C%=1)THEN PROCchu
k(" "+CR$,-1):GOTO650 ELSE PROCchuk(CR$,
-1):C%=1:GOTO650
  630   C%=0:PROCchuk(CHR$(A%),-1)
  650   NEXT:CLOSE#0:GOTO699
  660 IFY%=130THEN1NPUT"*" F$:S%=580+F$+C
HR$13:X%=&00:Y%=5:CALL&FFF7:GOTO699
  699 COLOUR128:COLOUR1:A%=138:X%=2:GOTO
540
  750 DEFPROCchuk(Q$,L%):FORI%=1TOLEN(Q$
):REPEAT:UNTIL(ADVAL(-3)<>0):A%=138:X%=2
:Y%=ASC(MID$(Q$,I%,1)):CALL&FFF4:IFL%=-1
THENPRINT;MID$(Q$,I%,1);
  760   NEXT:ENDPROC
  800 DEFPROCasm:FORY%=0TO3STEP2:P%=E%:t
OPTY%:.wait LDA#0:STA&FE6B:STAtemp2:.sit
 LDAlow,X:STA&FE64:LDAhigh,X:STA&FE65:LD
A#64:.d BIT&FE6D:BEQd:INCtemp2:LDAtemp2:
CMP#1:BEQsit:RTS
  820   .dial LDA#&C4:STA&FE4C:LDA#2:LDX
#2:JSR&FFF4:LDX#3:JSRwait:JSRwait:JSRwai
t:JSRwait:JSRwait:JSRwait:LDA#0:STAtemp:
.di LDXtemp:LDYstart:INCstart:LDAbuf,X:S
TA&7F98,Y:BMItail:AND#15:BNEP%+4:LDA#10:
CMP#1:BCCok:INCtemp:BNEdi
  825   .ok STAtemp1
  830   .lp LDA#&81:LDX#0:LDY#0:JSR&FFF4
:CPY#&FF:BEQlok:CPY#&1B:BNEP%+7:LDA#126:
JSR&FFF4:LDA#1:STAtemp2:LDA#2:LDX#0:JMP&
FFF4:.lok LDA#2:LDX#0:LDY#0:JSR&FFF4:LDX
#1:JSRwait
  835   LDA#2:LDX#2:JSR&FFF4:LDX#2:JSRwa
it:DECtemp1:LDAtemp1:BNElp:LDX#3:JSRwait
:JSRwait:INCtemp:JMPdi
  840   .tail LDX#2:JSRwait:LDA#0:STAtem
p2:CLI:RTS
  860   .trans LDA#1:STA&ZA0:STAtemp:.zi
 LDA#&00:LDX#&FD:JSR&FFF4:CPX#0:BNEzz:LD
A#13B:LDX#2:LDY#19:JSR&FFF4:.zz LDA#&81:
LDX#0:LDY#0:JSR&FFF4:CPY#&1B:BNEP%+10:LD
A#126:JSR&FFF4:JMPfail:CPY#&FF:BNEfail:L
DA&ZA0:BEQaffirm:LDA&FE40:AND#16:BNEz2
  865   LDA#15:LDX#0:JSR&FFF4:LDAtemp:BN
Etrans:INCtemp
  870   LDA#7:LDX#none MOD256:LDY#none D
IV256:JSR&FFF1:JMPtrans:.z2 LDAtemp:BEQz
1:LDA#0:STAtemp:LDA#7:LDX#one MOD256:LDY
#one DIV256:JSR&FFF1:JMPz1:.affirm LDA#1
:STAtemp:RTS:.fail LDA#0:STAtemp:RTS
  880   .one
  990   I:!P%=&FFF800:I:P%!4=&00140030:n
one=P%+8:P%!8=1:P%!12=&140000:high=P%+16
:P%!16=&FF400800:P%!20=&4074E8AC:low=P%+
20:buf=P%+24:NEXT:ENDPROC
  999 REM "Telephone numbers follow... u
se "-" only as spacer
 1000 DATA"Microweb","061-456-4157","Bla
ndford Board","0258-54494","CABB","01-63
1-3076"
 1010 DATA"TBBS London","01-348-9400","D
ISTEL","01-679-1888","NBBS East 11pm-7am
","0692-630610"
 1020 DATA"MOBB","061-736-8449"
 1999 DATA *,*
```

**Display Electronics, suppliers of the 2B modem are at: 32 Biggin Way, Upper Norwood, London SE19 3XF. Telephone: 01 679 4414.**

When the carrier is truly present, it will register as a constant high on the "carrier detect" line. Seeing this, the program will then drop through to the main editor. Note that pressing "ESCAPE" at any point aborts the call and returns you to the dialling menu. Distel is basically a simple-to-use window shopping program run by DE. You can use it like an electronic catalogue. It is perhaps not what built-in boards are all about but it really is hardly ever engaged!

## The editor

The editor displays all its output in Modeø, a 80 x 52 two colour mode. The background is VDU19'd into blue, and the foreground left white. This is purely personal preference though, and you can change the "mode" statement at line 530 to whatever you like – even mode 7.

The editor is usually transparent to the user; you type your letters and they get sent. There are a couple of special features on the function keys, but they will be dealt with later.

When talking to a bulletin board, keys pressed on the BBC are only sent to the RS423 driver – not to the screen driver. If that were all that happened, you'd be typing blind . . . but the remote computer "echoes" back each character as it gets it so you can see if what it received was the same as what you sent. Handling that is the main function of the editing part of the program, but it also includes three "utilities" on

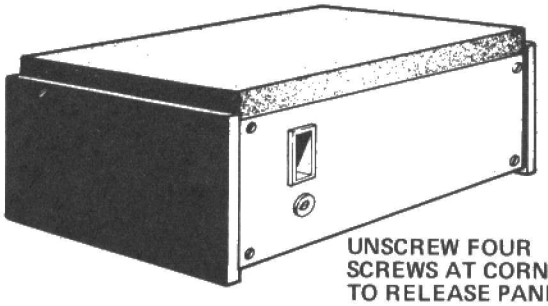



*Figure 2(a). Four screws on the rear panel of the 2B need to be removed to gain access to the inside.*

format it to the reader's terminal width. Any ASCII file should work; I have successfully tried Wordwise files. Note that the last character in the file must be a single carriage return. You can also use carriage returns before this in order to cut your text into neat "paragraph" sized blocks. After the file has all been sent there'll be a short pause while the BBC empties its buffer, then the normal "Edit/Continue/List/Abort/Save" prompt will appear.

If you list your message at this point, it'll look like the normal TBBS editor mess with lines that seem to end halfway through a paragraph etc. But if you Save the message and read it back as a message, everything has been neatly formatted.

f2 prints up a *, then waits for you to type a line. It then processes the line as it would a * command. Like, doing an f2 then typing

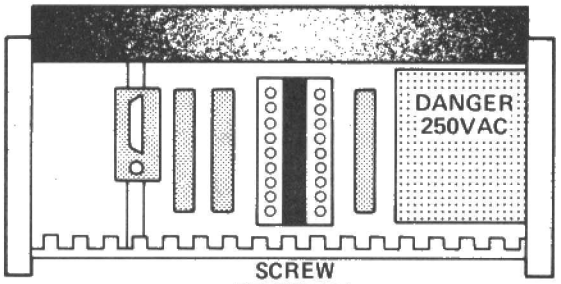



*Figure 2(b). The location of the terminal block within the 2B modem.*

F0 through 2.

The only thing F0 contains is your name. This is very useful when logging on, as all the boards want to know this. The string for this is set up on line 80, and the STRING$(20,CHR$0) between your first name and second name is necessary to make a pause while the Bulletin board asks "LAST NAME?"

The ASCII files are downloaded with F1. If you wordprocess questions or replies off-line (ie before 'phoning up), you can send the files toute suite using the "Block" entry mode on TBBS systems. Reply "N" to the "Prompts?" question from TBBS when doing this.

With this you don't have to worry about line endings, or putting a space between carriage returns, like you have to when using "Line" mode in TBBS messages because between the TBBS and the terminal program will tidy up your message and ,<return> would give you a *CAT.

Note that displays printed by the function keys comes out as reverse field print, ie blue on white, so you can see what's come from where.

File down/upload commands are to be implemented on other keys in the near future – as soon as I can find the protocol from somewhere!

## Auto answer?

**Table 1** gives the technical details of what got connected to where in the modem-computer cable. The status of the "Ring indicator" can be monitored by the value of (?&FE40)AND32, which will return 32 if the modem is being "rung up" and 0 if nothing much is happening. Likewise, the status of the "Carrier detect" can be determined by (?&FE40)AND16; 16 if the carrier is present and 0 if it isn't.

With 300 baud full duplex communications, which is what this modem is using, the Modem that is rung up is called the CALLing modem, and the the answering modem the (full marks!) ANSWERing modem. If a modem is the caller, it has to use a different set of tones to that of the answerer. The 2B is told whether it is calling or answering by a signal line called "Select Transmit Frequency". This is connected to CB2 of the Beeb's internal VIA, calling ?&FE4C=&C4, if answering, ?&FE4C=&E4.

The signal line "DTR", or "Data Terminal Ready" is connected to the BBC's RTS output; *FX2,2 or *FX2,1 makes this line active, while BREAK or a *FX2,0 makes it inactive. While active, the modem is connected to the telephone line.

For Auto Answer, the sequence of events is as follows:

*MONITOR the Ring detect line until it becomes true ((?&FE40)AND32 =32). This indicates someone is ringing up.

*Set the modem's transmit frequency to the "answer" pair by doing a ?&FE4C=&E4

*"Pick up the phone" by doing a *FX2,2, then pause for two seconds. The pause gives time for the Telecom exchanges to realise you've picked the 'phone up and connect you to the caller.

*Monitor the "Carrier Detect" line for 15 seconds. If within that period it doesn't become true (ie (?&FE40)AND16 =16) for two sequential seconds then do a *FX2,0 to put the 'phone down, and return to the first step to wait for a new call.

## PARTS LIST

2 5V1 BZY88C zener diodes
2 IN4148 diodes
2 1k2 ⅛W resistors
1 25 way D plug + cover
1 15 way D plug + cover
1 5 pin 'Domino' DIN plug (= plug for BBC's RS423 port)
1 metre 8-core cable
1 Telecom Plug to 4 spade connector lead

### AUTO ANSWER ONLY

1 1μF 250V **UNPOLARISED** Capacitor
1 TH3 (eqvt VA1026 or CZ13) Rod thermistor

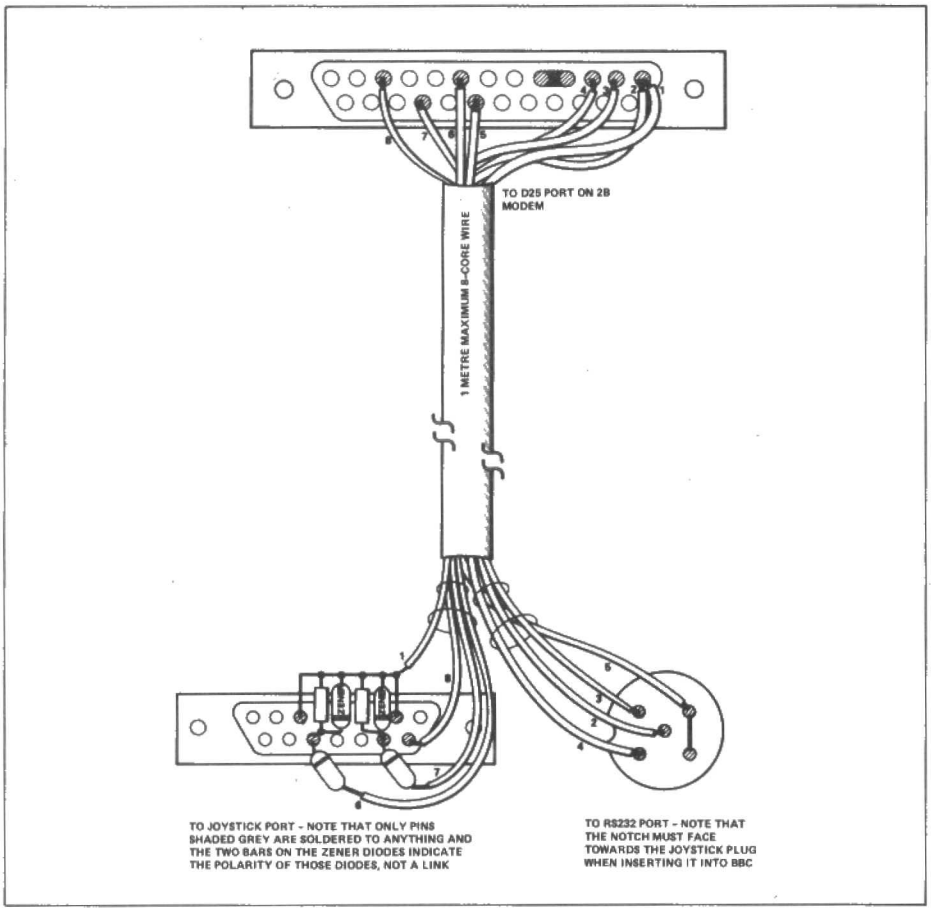



*Figure 1. Details of the connecting lead and components necessary to link the 2B and BBC micro.*

*If the carrier detect line did go high for the two-second period, start transmitting your stuff. The algorithm needs two seconds of carrier to be sure it doesn't respond to line noise.

While transmitting your stuff, check periodically that the Carrier of the calling modem is still present; if it isn't then the guy has rung off, and you should return to the first step to await the next caller.

## Phone bill hints

Here is some sympathetic help for sufferers of hefty 'phone bills.

1. Limit yourself to one board a day, maximum. Choose just four or five boards to frequent, and do just one a day in strict rotation. This will improve the quality of your hacking, thus more people will have had time to reply by the time you get around to that board again.

2. If possible, dial in very late at night or very early in the morning, ie 5am. Otherwise the boards will be all engaged.

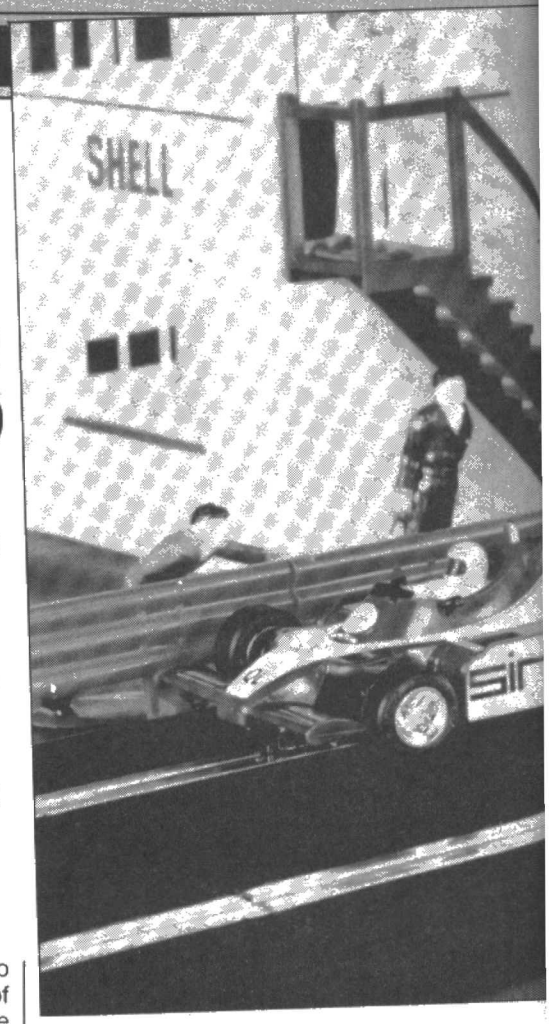
I'm currently working on a complex bulletin board for the BBC with one of these 2B auto-answer modems. You'll also need two double-sided 80k discs with a double density DOS (eg OPUS DDOS).

It'll be much friendlier than TBBS and with a clock that keeps the right time! If you are thinking of setting up a Board, please drop me a line c/o *E&CM* with any suggestions/comments.

# LAPPING UP MICRO SALES

## WHO'S ON THE RIGHT TRACK?

**Michael Graham ponders the future of the UK micro computer industry in the light of recent trends both in this country and in the USA.**




During the last quarter of 1984 the micro-computer industry indulged in the orgy of promotional spending and accompanying hype that has, for the past few years, been associated with the run up to the Christmas holiday. A number of different factors would seem to indicate, though, that Christmas past may be the last occasion to witness such a concentrated marketing effort from the various major computer manufacturers.

For some time now many people in the computer industry have seen the frenzied activity during the last few months of every year as being unhealthy for the industry as a whole. Some notable firms have built up such high hopes of a Christmas bonanza that the success, or failure of entire companies have hinged on achieving a significant slice of the festive market. While some industries, although the firework trade is the only one to spring to mind here, seem to exist on a brief period of sales followed by months of inactivity, the computer industry would seem ill-equipped to cope with such a pattern of sales. Analysis of the sales patterns in countries such as Japan indicates that the marketing men may have been the authors of their own misfortune. If a company commits a disproportionate amount of advertising monies to the few months preceding Christmas it would not seem unreasonable to expect that sales during this period will show a dramatic upturn.

Christmas '84 saw Acorn commit £4.5m to promoting the Electron and BBC micros while Sinclair were busily shipping 20,000 computers per day from their Camberley distribution depot. Commodore had announced during the late summer that they were to commit more cash than any of their competitors in the battle for the spending money of the Christmas micro buyer. The people on the receiving end of all this cash set aside for promotion were not, as in previous years, the specialist computer press. Although this section of the media, in which *E&CM* must be included, benefitted from the increased activity, only a fraction of budgets were allocated to this sector and by far the majority of available funds were spent with TV and the national daily and Sunday press. 1984 was the year that the micro-computer became just another consumer product. The novelty and specialist nature of computers came to an end.

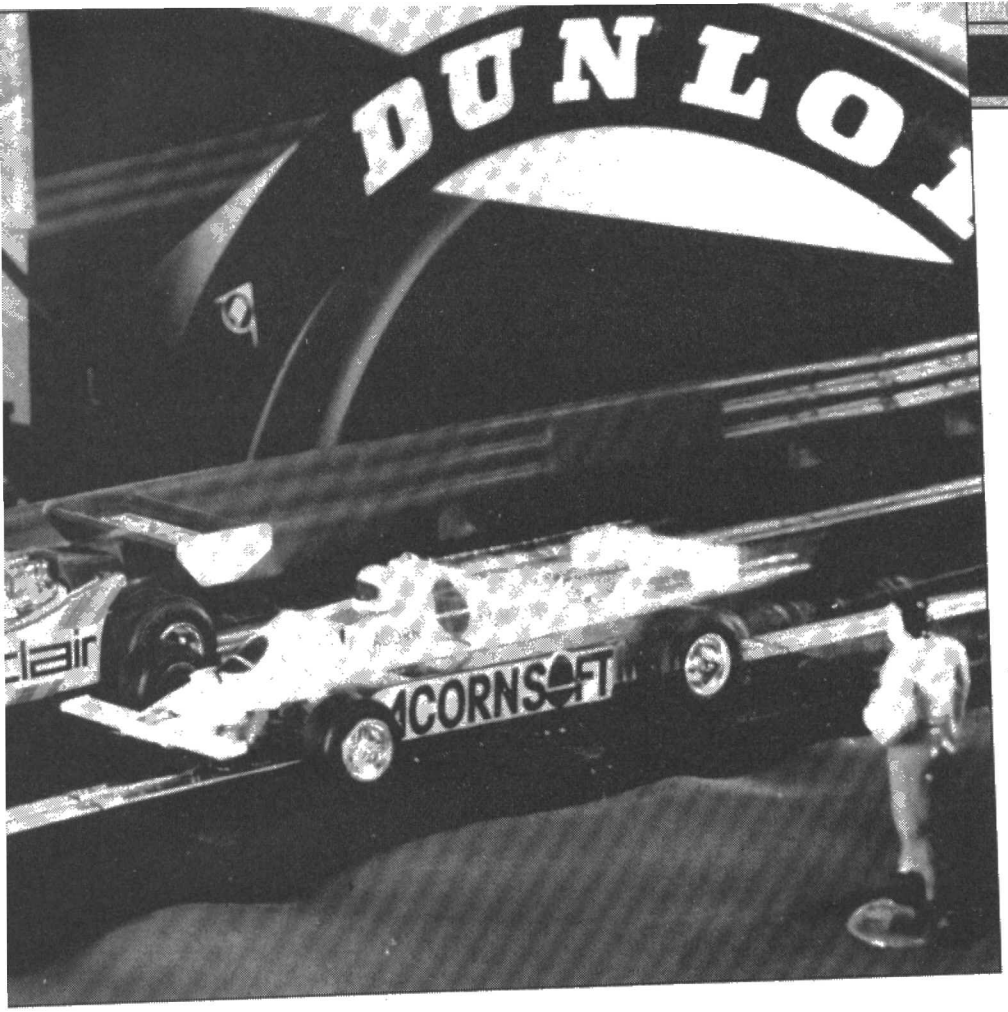
**"The frenzied activity during the last few months of every year is unhealthy for the industry as a whole".**

At this point it seems a good idea to introduce the American experience into the equation. While the US market is in many ways different to that in this country, it is worth remembering that the policy the States is following today, we'll more than likely be following tomorrow.

Last year, Christmas 1983, the US market witnessed a price war at the low cost end of the micro market and this saw the end of more than one major micro manufacturer. Commodore, in a slimmed down form are the only major force to survive the effects of last year's suicidal encounters. Radio Shack (Tandy) still relying on the ever-ageing CoCo, Atari (with heavily discounted hardware) and the Coleco Adam, were the only other participants at the home end of the market this year and each of them ended up as 'also-rans'. Commodore, having weathered the storms of last year were there to reap the benefits of being the only fish in the sea.

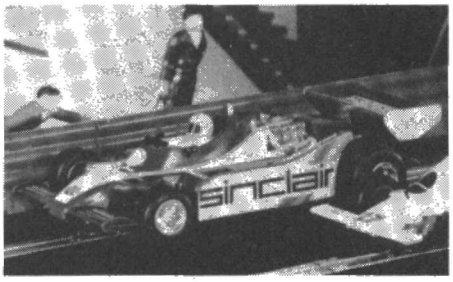
The American market is now, compared to only a short time ago, quite mature. Computers must compete with the range of other household goods for the available cash in circulation. They must compete on value for money rather than that of novely. The market has very firmly divided into those people purchasing a micro for strictly home (usually entertainment use) and those buying for more serious (business) orientated use. The first group are looking for the best value for money, this encompasses not only low cost hardware but also a large amount of keenly priced software and it is no wonder that the Commodore 64 has faired so well in this market. The more serious purchaser is looking for a performance level that supports the latest sophisticated business software and possibly for hardware that will be compatible with the computer at work. These buyers are turning to the likes of the Apple Macintosh and the IBM PC jnr.

In this country the situation is somewhat different. The likes of the IBM and Apple machines still attract a price tag that is significantly above the level of that which




many small business people are prepared to pay and, in contrast to the States, there are a number of companies offering relatively low cost hardware that is capable of series application. Not least of these are the likes of Acorn and Sinclair. Acorn have, in the BBC micro, a computer that is starting to show its age but, by virtue of the numerous add-ons that have been developed for it, it is capable of meeting the demands of many business applicatons. Also Sinclair, in the shape of the QL, have a machine that, in offering the power of a 16-bit processor, may have something to offer the more serious end of the market.

Both machines are however somewhat flawed. In the case of the BBC micro the problem is that to equip the computer with the hardware and software necessary to bring it to the performance level required of a 'serious' computer requires many hundreds of pounds worth of expenditure. Memory is insufficient, so add a second processor; the computer needs disk drives




and a monitor, perhaps there is a demand for Prestel/Viewdata capability – it all costs money and lots of it. In the case of the QL the major problem is one of image. At first sight the look of the machine does little to instill confidence in the serious user and the micro drives, while being a low cost form of mass storage, by all accounts are just not reliable enough for consideration in demanding applications.

The first signs that the pattern of microcomputer sales are about to change in this country are already with us. The idea that hardware can appeal both to the home (games) buyer in addition to the serious (business) user, is starting to recede. Dixons, a High Street store not readily identified with the image of the businessman looking for advice on the purchase of his first computer system, has started to advertise the likes of the Apricot F1 alongside the Amstrad CPC64, the 16-bit Sanyo MBC555 next to the Commodore 64. Even a short time ago these 'upmarket' items of hardware would have seemed out of place in the window of a High Street multiple. Today though, in terms of price v performance this sort of computer is appealing to the more serious end of the market. It may not be too long before the PCs of this world make their High Street debut.

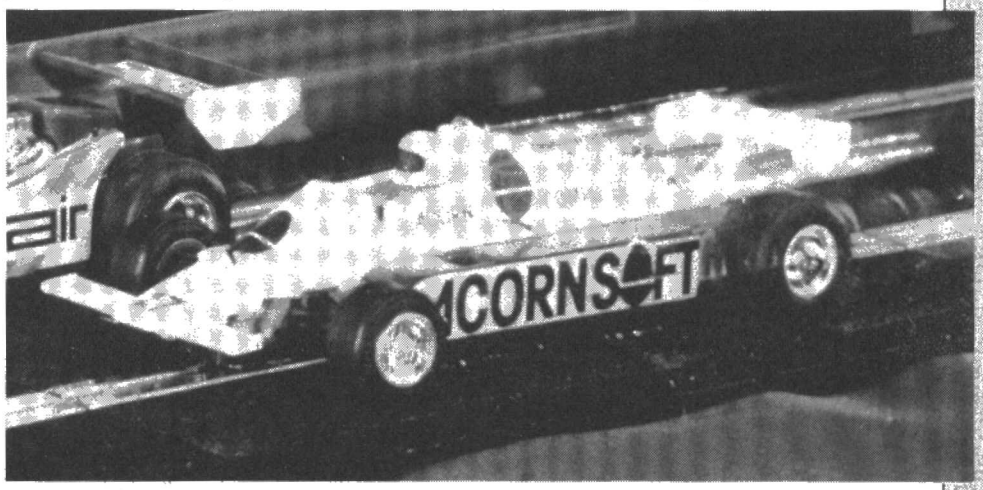
The somewhat troubled Advance has after all made an appearance in W H Smith,

**"The pattern of microcomputer sales is about to change in this country".**

and while the reason for that move may not have its roots in the soundest of commercial judgement, the principle was established.

All this tends to suggest that the pattern of the race for festive season sales could see a number of significant changes in 1985. In the first place the race could turn out to be more of a crawl with an altogether welcome levelling off of microcomputer sales as purchasers buy, not on the spur of the moment, but after serious consideration of their needs. A trend towards a greater gap between the lower and more serious ends of the market which could see the likes of the BBC micro and, possibly the relatively expensive, MSX computers caught in a no-man's-land, appealing neither to the cost conscious games buyer nor the potential business purchaser.

A further hardening of the market is indicated by recent figures that show that only 13% of those people without a computer intend to buy within the near future. Contrast this with findings from the same survey that show that of those who own a computer 37% of home users and a massive 52% of business users intend to upgrade in the next year.




The computer buying public has come of age very rapidly; the question is will the industry manage to do likewise? This year will see more than one manufacturer in difficulty if they do not pay careful attention to the needs of an ever more discerning market place, one that is rapidly dividing itself into two groups with each knowing what it wants and being determined to get it.

# MAKING THE RIGHT CONNECTIONS

**Connecting computer systems via RS232 lines is well known to cause many frustrating problems. Both hardware and software incompatibility can transform such an exercise into a nightmare. Fortunately anyone reading Adam Denning's latest article will be able to rest easy – he's done all the hard work.**

In the past few months we have delved into QL machine code in such detail that now almost anything is possible. The next step is inevitably hardware, what we've got, what we need and what we can do.

The most important interfaces available on the QL are the RS-232 ports, ser1 and ser2. The common application for these is of course a printer, but there is a host of other devices which we can connect to the QL via one of these serial ports. For anybody fortunate enough to have been given a Sinclair RS232 lead free with the machine things are relatively simple. If you didn't get one then the options are either to buy one – but Sinclair is charging a fairly high price – or to build one up. As the connections are detailed in the Concepts section of the User Guide this is a simple process, and can be done cheaply if you can get hold of the requisite British Telecom plug. There are shops here and there which sell them but make sure you buy the rights ones – the normal BT phone jack is not quite the same.

The connections needed to make an (almost!) standard RS-232C lead are shown in **Figure 1.** This is suitable for connection to ser1, as ser2 is wired in a slightly different way and turns out to be rather less useful than ser1. With this lead or the Sinclair Research lead, the QL can be plugged directly into most machines and data transferred without further ado. So, although purists would day that the QL's RS232 is not completely standard, these connections are enough to make it work perfectly with all Apricots (and we've tried them), the IBM PC, all the Brother machines and almost any other machine sporting a serial 25 pin D socket.

Two common machines with which the QL will not immediately communicate are the Spectrum and the BBC micro. The first because it doesn't have an RS232 port and the second because its manufacturer has decided to follow a different standard – RS423. The addition of Interface 1 to a Spectrum goes part way to solving the communication problem, but further difficulties arise as the Interface 1 RS232 is

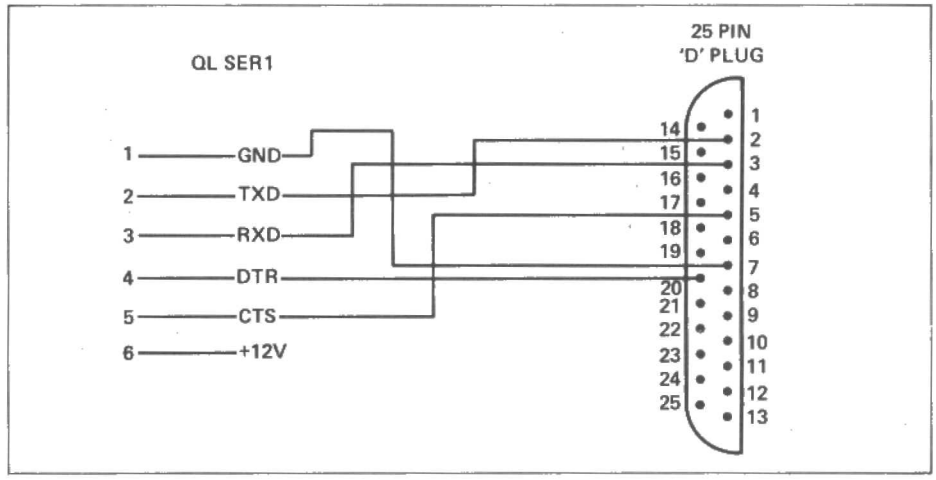



*Figure 1. QL RS232 connectors.*

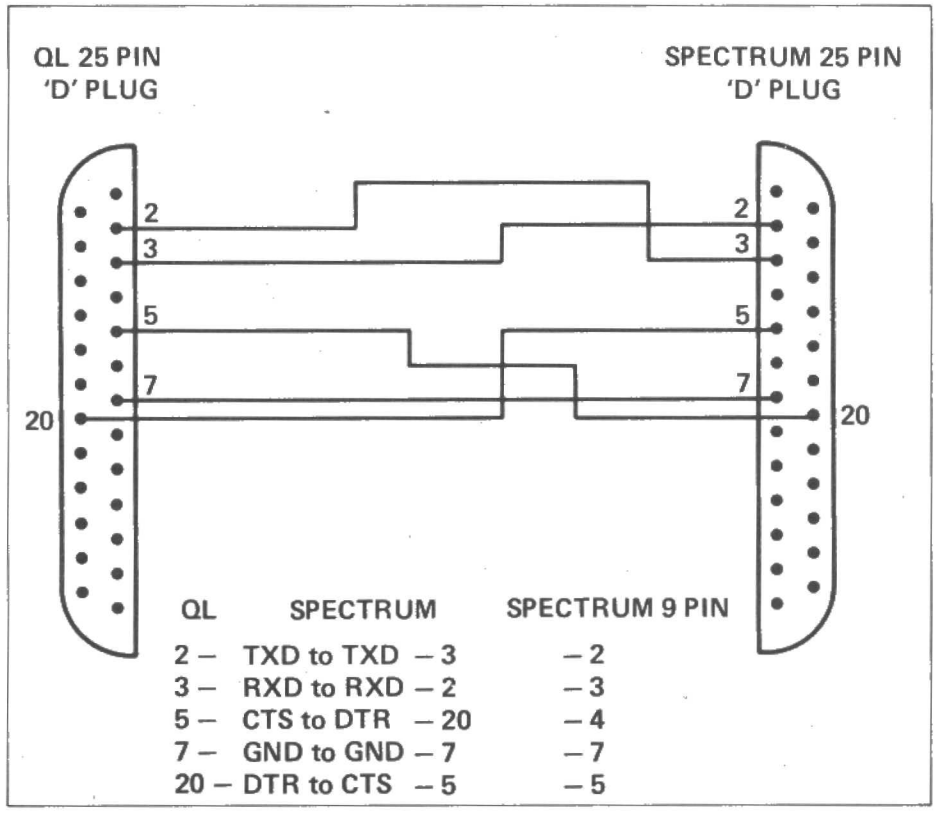



*Figure 2. QL to Spectrum RS232 connectors.*

wired oddly and terminates in a 9 pin D socket. Again, a lead can be purchased from Sinclair Research to aid connection, but the orientation of a couple of wires will probably need to be changed.

Connection to a BBC Micro is a little more interesting as it requires an RS-232 to RS423 interface. The former normally operates using levels of + and −12V while the latter uses 5V levels, so at first sight things look bad. However, the RS-232 standard is defined in such a way that logic levels of 0 and 1 occupy defined ranges rather than absolute values, so the QL should be capable of understanding what the BBC is sending to it. For the same reason, the BBC turns out to be able to understand what the QL sends to it. As both micros set their default baud rates to 9600, it's almost a 'plug in and go' situation, except that at this rate of transmission the QL needs more than one stop bit to receive RS-232 data correctly. The BBC defaults to one stop bit, so we'll have to program the 6850 ACIA inside the BBC Micro to send the information in the correct format. The versatility of the OSBYTE operating system call (*FX commands) makes this a very simple process. But first, let's examine the connections required. They're shown in **Figure 3.**

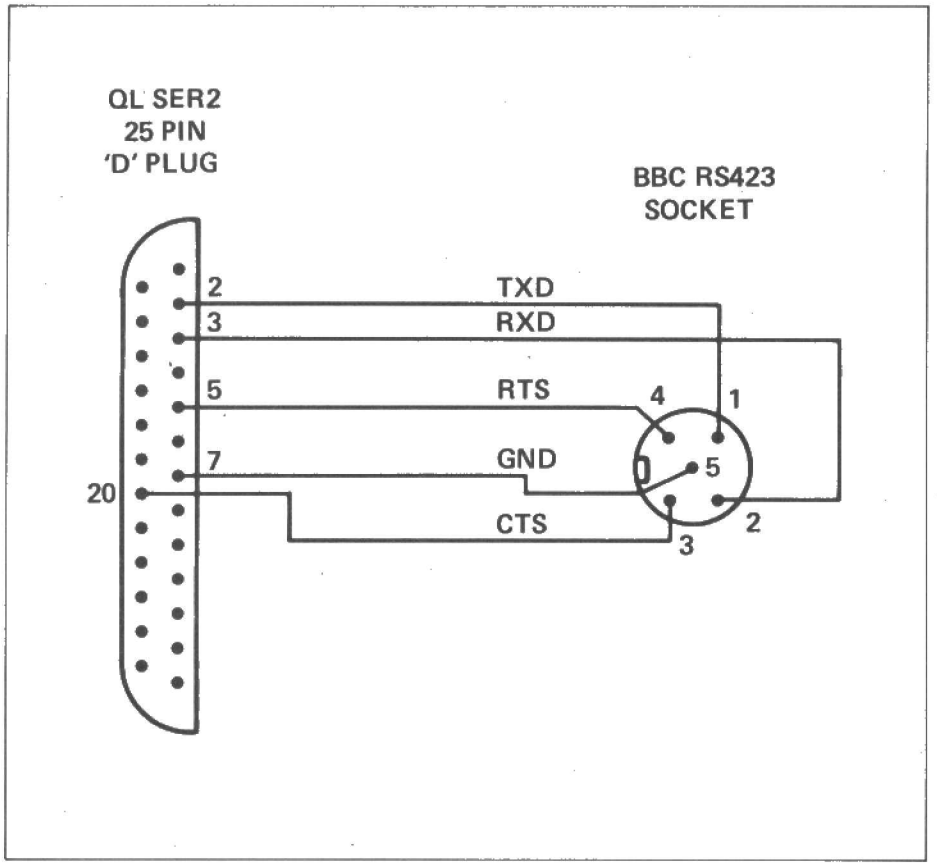



*Figure 3. QL to BBC connectors.*

As universal RS-232 connections are so useful we built up a little circuit on veroboard which allows the connection of a QL to a BBC, a Spectrum or another QL. As a side effect it also allows connection of a Spectrum to a BBC and a BBC to a modem or other serial device. It also transpires that by connecting the QL to a BBC Micro and setting the Quill printer drivers up correctly, we can use the BBC Micro as a fast printer spooler for Quill. This is especially useful as the Centronics parallel port of the BBC

**LISTING 1. QL terminal.**

```
100 OPEN£3,ser1c
110 REPeat loop
120   REPeat loop1
130     a$=INKEY$:IF a$<>'':EXIT loop1
140     a$=INKEY$(£3):IF a$='':NEXT loop1
150     PRINT a$;
160   END REPeat loop1
170   REPeat loop2
180     PRINT£3;a$;
190     PRINT a$;
200     a$=INKEY$:IF a$='':EXIT loop2
210   END REPeat loop2
220 END REPeat loop
```

**LISTING 2. Spectrum terminal.**

```
10 FORMAT "b";9600
20 OPEN£4; "b"
30 LET a$=INKEY$: IF a$<>"" THEN GO TO 70
40 LET a$=INKEY$£4: IF a$="" THEN GO TO 30
50 PRINT a$;
60 GO TO 30
70 PRINT£4;a$;
80 PRINT a$;
90 LET a$=INKEY$: IF a$="" THEN GO TO 30
100 GO TO 70
```

**LISTING 3. BBC Micro terminal.**

```
10 *FX8,7
20 *FX7,7
30 *FX156,146,0
40 *FX2,0
50 A$=INKEY$(0):IF A$<>"" THEN 100
60 *FX2,1
70 A$=INKEY$(10):IF A$="" THEN 40
80 PRINT A$;
90 GOTO 40
100 *FX3,1
110 PRINT A$;
120 *FX3,0
130 A$=INKEY$(0):IF A$="" THEN 50
140 GOTO 100
```

**LISTING 4. BBC Quill Spooler.**

```
10 ONERRORGOTO150
20 *FX8,7
30 *FX7,7
40 *FX156,146,0
50 *FX15,0
60 INPUT"Filename: "F$
70 MODE3
80 S%=OPENOUT(F$)
90 *FX2,1
100 *FX15,0
110 REPEAT A%=GET
120 IF A%=10 OR A%=13 OR A%>31 THEN VDUA%
130 IF A%<>26 AND A%<>9 THEN BPUT£S%,A%
140 UNTIL A%=26
150 CLOSE£S%
160 *FX2,0
170 MODE7
```

**LISTING 5. Quill Printer Driver.**

Just ensure that data is sent to ser1c and that no printer code sends CTRL-Z, as this will close the file.

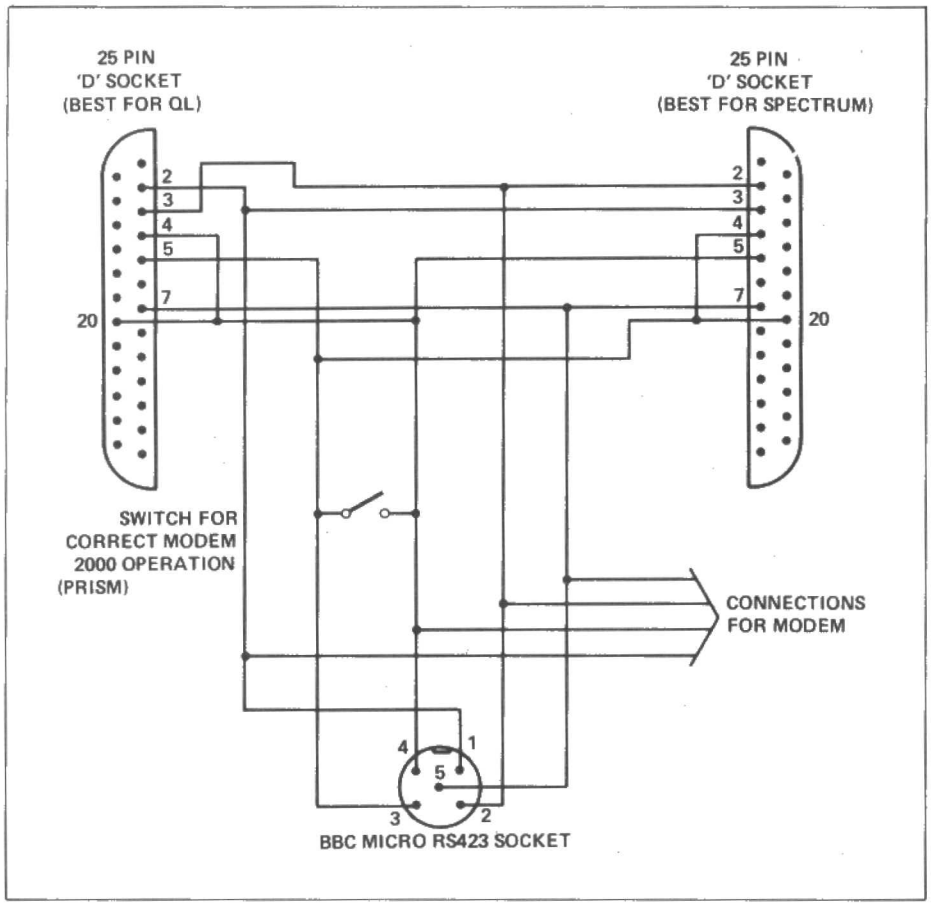



*Figure 4. Circuit diagram for universal interface.*

Micro seems to be a lot faster at printing out data to an Epson RX80 than the QL through a proprietary serial to parallel converter.

The circuit diagram for the 'universal interface' is shown in **Figure 5.** Short and sweet terminal programs for the BBC, the Spectrum and QL are shown below along with a program which allows the BBC to act as a Quill printer spooler and the requisite printer driver for Quill and an Epson printer.

I'm afraid the last two programs are rather unstructured – the first because Spectrum BASIC does not support REPEAT and so on and the second because BBC BASIC's REPEAT loops are not as versatile as SuperBASIC's. In the BBC example, we use *FX156 to alter the date sent out by the 6850 to eight bits with two stop bits. Note that the INKEY$ time-out in line 70 may need alteration – 10 works fine here, but apparently any value from 5 to 50 will suffice. Obviously the smaller the better.

To make the BBC act as a printer spooler for Quill we need to use another program, shown below. This prompts for a suitable filename under which to save the printed document on the BBC, and then proceeds to read all RS423 data into that file. Notice that we ensure that the QL sends its data along **ser1c,** so that when the end of the Quill file is reached it sends an implicit CTRL-Z to close the file at the Beeb's end.

# BBC PRINTER BUFFER PART TWO

## The software for Brian Alderwick and Peter Simpson's BBC printer buffer that turns any single bank of sideways RAM into a versatile printer buffer.

```
   10 REM*********************************
   20 REM                                 *
   30 REM SIDEWAYS RAM PRINTER BUFFER     *
   40 REM                                 *
   50 REM      by P W G SIMPSON AND       *
   60 REM                                 *
   70 REM         B V ALDERWICK           *
   80 REM                                 *
   90 REM*********************************
  100
  110 IF TOP>&6910 THEN PRINT"NOT ENOUGH SPA
CE":END
  120 osrdch=&FFE0
  130 osasci=&FFE3
  140 osnewl=&FFE7
  150 oswrch=&FFEE
  160 osbyte=&FFF4
  170 oscli=&FFF7
  180 romselect=&FE30
  190 privwp=&DF0
  200
  210 work=&A8
  220 workl=&A9
  230 myno=&AA
  240 ramno=&AB
  250
  260 read=&880
  270 write=&883
  280 base=&8A0
  290 pil=&FFFF
  300 pih=&FFFF
  310 pol=&FFFF
  320 poh=&FFFF
  330 bef=base + 0
  340 bff=base + 1
  350 bs=base + 2
  360 bsl=base + 2
  370 bsh=base + 3
  380 bf=base + 4
  390 bfl=base + 4
  400 bfh=base + 5
  410 insvec=&8B0
  420 remvec=&8B2
  430 cntvec=&8B4
  440 tempa=&8B6
  450 tempx=&8B7
  460 tempy=&8B8
  470 purgeflag=&8B9
  480
  490 version$="1.00"
  500 copy$="(C) P W G SIMPSON and "
  510 copy$=copy$+"B V ALDERWICK 1984"
  520 name$="SIDEWAYS RAM PRINTER BUFFER"
  530
  540 code%=&7100
  550 FOR I%=4 TO 7 STEP 3
  560 O%=code%
  570 P%=&80C0
  580 [         OPT I%
  590 .romstart BRK
  600           BRK
  610           BRK
  620           JMP start
  630
  640           EQUB &82
  650           EQUB copy-&8000
  660           EQUB 0
  670           EQUS name$
  680           EQUB 0
  690           EQUS version$
  700 .copy     EQUB 0
  710           EQUS copy$
  720           EQUB 0
  730
  740 .start    PHP
  750           CMP #2
  760           BNE sta1
  770           JMP private
  780
  790 .sta1     CMP #4
  800           BNE sta2
  810           JMP comm
  820
  830 .sta2     CMP #9
  840           BNE sta3
  850           JMP help
  860
  870 .end      PLA
  880           TAY
  890           PLA
  900           TAX
  910           PLA
  920 .sta3     PLP
  930           RTS
  940
  950 .private  PHA
  960           TXA
  970           PHA
  980           TYA
  990           PHA
 1000           LDA privwp,X
 1010           AND #&80
 1020           BEQ priv1
 1030           STX myno
 1040           JSR convecs
 1050           JSR purgesub
 1060 .priv1    JMP end
 1070
 1080 .endz     PLA
 1090           TAY
 1100           PLA
 1110           TAX
 1120           PLA
 1130           PLP
 1140           LDA #0
 1150           RTS
 1160 \
 1170 \ CODE FOR *BUFFON COMMAND
 1180 \
 1190 .buffon   JSR getxy
 1200           STX myno
 1210           LDA privwp,X
 1220           BEQ bu2
 1230           JSR print
 1240           EQUS "Buffer already "
 1250           EQUS "active"
 1260           EQUB 0
 1270           JSR osnewl
 1280           JMP endz
 1290
 1300 .bu2      LDX #codeend-codeloc
 1310 .bu3      LDA codeloc,X
 1320           STA &100,X
 1330           DEX
 1340           BPL bu3
 1350           JSR &100
 1360           CPX #&FF
 1370           BNE bu4
 1380           JSR print
 1390           EQUS "No sideways RAM "
 1400           EQUS "present!"
 1410           EQUB 0
 1420           JSR osnewl
 1430           JMP abend
 1440
 1450 .cnpv     JMP (&22E)
 1460
 1470 .bu4      STX ramno
 1480           LDX #3
 1490           CLV
 1500           CLC
 1510           JSR cnpv
 1520           TXA
 1530           BEQ bu5
 1540           JMP binuse
 1550
 1560 .bu5      TYA
 1570           BEQ bu6
 1580           JMP binuse
 1590
 1600 .bu6      LDX #subend-substart
 1610 .bu7      LDA substart,X
 1620           STA &880,X
 1630           DEX
 1640           BPL bu7
 1650           LDX ramno
 1660           STX rno
 1670           CPX myno
 1680           BNE bu8
 1690           JMP meram
 1700
 1710 .bu8      LDA #252
 1720           LDX #0
 1730           LDY #&FF
 1740           JSR osbyte
 1750           CPX ramno
 1760           BNE bu9
 1770           JSR print
 1780           EQUS "Sideways RAM in use "
 1790           EQUS "by current language"
 1800           EQUB 0
 1810           JSR osnewl
 1820           JMP abend
 1830
 1840 .bu9      LDA #170
 1850           LDX #0
 1860           LDY #&FF
 1870           JSR osbyte
 1880           STX work
 1890           STY workl
 1900           LDY ramno
 1910           LDA (work),Y
 1920           BEQ ramfnd
 1930           JSR print
 1940           EQUS"RAM already in use! "
 1950           EQUS "Continue (Y/N)? "
 1960           EQUB 0
 1970           JSR osrdch
 1980           BCC bu20
 1990           LDA #126
 2000           JSR osbyte
 2010           JSR print
 2020           EQUS "Escape"
 2030           EQUB 0
 2040           JSR osnewl
 2050           JMP endz
 2060
 2070 .bu20     JSR oswrch
 2080           PHA
 2090           JSR osnewl
 2100           PLA
 2110           AND #&DF
 2120           CMP #ASC"Y"
 2130           BEQ ramfnd
 2140 .abend    JSR print
 2150           EQUS "Command aborted"
 2160           EQUB 0
 2170           JSR osnewl
 2180           JMP endz
 2190
 2200 .ramfnd   LDA #&80
 2210           BNE rf2
 2220 .meram    LDA #&80 + ((romend-romstart
) DIV 256) + 1
 2230 .rf2      STA pih
 2240           STA bsh
 2250           LDY #0
 2260           STY pil
 2270           STY pol
 2280 .rf3      LDA #&AA
 2290           JSR write
 2300           INC pih
 2310           LDA pih
 2320           CMP #&C0 \ MAX RAM
 2330           BNE rf3  \ SIZE
 2340           LDA bsh
 2350           STA pih
 2360           STA poh
 2370           LDA #&55
 2380           JSR write
 2390 .rf4      INC poh
 2400           JSR read
 2410           CMP #&55
 2420           BEQ rf5
 2430           CMP #&AA
 2440           BEQ rf4
 2450 .rf5      LDA poh
 2460           STA bfh
 2470           LDA bsh
 2480           STA pih
 2490           STA poh
 2500           LDA #0
 2510           STA bfl
 2520           STA bsl
 2530           STA pil
 2540           STA pol
 2550           STA bff
 2560           LDA #1
 2570           STA bef
 2580           JSR convecs
 2590           LDX myno
 2600           LDA #&80
 2610           STA privwp,X
 2620           JSR print
 2630           EQUS "Buffer activated"
 2640           EQUB 0
 2650           JSR osnewl
 2660           JMP poff2
 2670
 2680 .convecs  LDX #5
 2690 .conv2    LDA &22A,X
 2700           STA insvec,X
 2710           DEX
 2720           BPL conv2
 2730           LDX #8
 2740 .conv3    LDA vecadd,X
 2750           STA &DDE,X
 2760           DEX
 2770           BPL conv3
 2780           LDA myno
 2790           STA &DE0
 2800           STA &DE3
 2810           STA &DE6
 2820           SEI
 2830           LDX #5
 2840 .conv4    LDA extadd,X
 2850           STA &22A,X
 2860           DEX
 2870           BPL conv4
 2880           CLI
 2890           RTS
 2900
 2910 .vecadd   EQUW inscode
 2920           EQUB 0
 2930           EQUW remcode
 2940           EQUB 0
 2950           EQUW cntcode
 2960           EQUB 0
 2970
 2980 .extadd   EQUW &FF3F
 2990           EQUW &FF42
 3000           EQUW &FF45
 3010 \
 3020 \ CODE FOR *BUFFOFF COMMAND
 3030 \
 3040 .buffoff  JSR getxy
 3050           STX myno
 3060           LDA privwp,X
 3070           BNE bo2
```

```
3080 .bnamess  JSR print
3090           EQUS "Buffer not active"
3100           EQUB 0
3110           JSR osnewl
3120           JMP endz
3130
3140 .bo2      LDA bef
3150           BNE bo3
3160 .binuse   JSR print
3170           EQUS "Buffer not empty"
3180           EQUB 13
3190           EQUB 0
3200           JMP abend
3210
3220 .bo3      LDA #0
3230           STA privwp,X
3240           LDX #5
3250 .bo4      LDA insvec,X
3260           STA &22A,X
3270           DEX
3280           BPL bo4
3290           JSR print
3300           EQUS "Buffer de-activated"
3310           EQUB 0
3320           JSR osnewl
3330           JMP endz
3340 \
3350 \ CODE FOR *PURGEON COMMAND
3360 \
3370 .purgeon  JSR getxy
3380           LDA privwp,X
3390           BNE pon2
3400           JMP bnamess
3410
3420 .pon2     SEC
3430           ROR purgeflag
3440           JSR print
3450           EQUS "ESCAPE will purge "
3460           EQUS "print buffer"
3470           EQUB 0
3480           JSR osnewl
3490           JMP endz
3500 \
3510 \ CODE FOR *PURGEOFF COMMAND
3520 \
3530 .purgeoff JSR getxy
3540           LDA privwp,X
3550           BNE poff2
3560           JMP bnamess
3570
3580 .poff2    CLC
3590           ROR purgeflag
3600           JSR print
3610           EQUS "ESCAPE will not "
3620           EQUS "purge print buffer"
3630           EQUB 0
3640           JSR osnewl
3650           JMP endz
3660 \
3670 \ CODE FOR *PURGE COMMAND
3680 \
3690 .purge    JSR getxy
3700           LDA privwp,X
3710           BNE pur2
3720           JMP bnamess
3730
3740 .pur2     JSR purgesub
3750           JSR print
3760           EQUS "Printer buffer purged"
3770           EQUB 0
3780           JSR osnewl
3790           JMP endz
3800
3810 .savereg  STA tempa
3820           STX tempx
3830           STY tempy
3840           RTS
3850
3860 .getreg   LDA tempa
3870           LDY tempy
3880 .getreg2  LDX tempx
3890           RTS
3900
3910 \
3920 \ CODE FOR BUFFER INSERTION
3930 \
3940 .inscode  PHP
3950           SEI
3960           CPX #3
3970           BEQ ins2
3980           PLP
3990           JMP (insvec)
4000
4010 .ins2     JSR savereg
4020           LDA bff
4030           BEQ ins3
4040           LDA tempa
4050           PLP
4060           SEC
4070           RTS
4080
4090 .ins3     LDA tempa
4100 \ STORE CHARACTER AT PI
4110           JSR write
4120 \ INCREMENT PI
4130           INC pil
4140           BNE ins4
4150           INC pih
4160 \ IF PI=BF THEN PI=BS
4170 .ins4     LDA pil
4180           CMP bfl
4190           BNE ins5
4200           LDA pih
4210           CMP bfh
4220           BNE ins5
4230           LDA bsl
4240           STA pil
4250           LDA bsh
4260           STA pih
4270 \ IF PI=PO THEN SET BUFF FULL FLAG
4280 .ins5     JSR pieqpo
4290           BNE ins6
4300           LDA #1
4310           STA bff
4320 \ CLEAR BUFF EMPTY FLAG
4330 .ins6     LDA #0
4340           STA bef
4350           PLP
4360           CLC
4370           JMP getreg
4380 \
4390 \ CODE FOR BUFFER REMOVAL
4400 \
4410 .remcode  PHP
4420           SEI
4430           CPX #3
4440           BEQ rem2
4450           PLP
4460           JMP (remvec)
4470
4480 .rem2     JSR savereg
4490           LDA bef
4500           BEQ rem3
4510           LDA tempa
4520           PLP
4530           SEC
4540           RTS
4550
4560 .rem3     BVC rem4
4570           JSR read
4580           TAY
4590           PLP
4600           CLC
4610           JMP getreg2
4620
4630 \ GET CHARACTER FROM BUFFER
4640 .rem4     JSR read
4650           PHA
4660 \ INCREMENT PO
4670           INC pol
4680           BNE rem5
4690           INC poh
4700 \ IF PO=BF THEN PO=BS
4710 .rem5     LDA pol
4720           CMP bfl
4730           BNE rem6
4740           LDA poh
4750           CMP bfh
4760           BNE rem6
4770           LDA bsl
4780           STA pol
4790           LDA bsh
4800           STA poh
4810 \ IF PI=PO THEN SET BUFF EMPTY FLAG
4820 .rem6     JSR pieqpo
4830           BNE rem7
4840           LDA #1
4850           STA bef
4860 \ CLEAR BUFFER FULL FLAG
4870 .rem7     LDA #0
4880           STA bff
4890 \ GET CHARACTER FROM STACK
4900           PLA
4910           TAY
4920           PLP
4930           CLC
4940           JMP getreg2
4950 \
4960 \ CODE FOR BUFFER PURGE AND COUNT
4970 \
4980 .cntcode  PHP
4990           CPX #3
5000           BEQ cnt2
5010           PLP
5020           JMP (cntvec)
5030
5040 .cnt2     PLP
5050           BVC cnt4
5060           JSR getxy
5070           PHP
5080           BIT &FF    \ TEST FOR
5090           BPL cnt3   \ ESCAPE PURGE
5100           BIT purgeflag
5110           BMI cnt3   \ TEST WHETHER
5120           PLP \ PURGE ON ESCAPE IS
5130           RTS \ ACTIVE
5140
5150 .cnt3     PLP
5160 \
5170 \ SUBROUTINE TO PURGE SIDEWAYS RAM
5180 \ BUFFER
5190 \
5200 .purgesub LDA bsl \ RESET VECTORS TO
5210           STA pil \ BS
5220           STA pol
5230           LDA bsh
5240           STA pih
5250           STA poh
5260           LDA #0  \ CLEAR BUFF FULL
5270           STA bff \ FLAG
5280           LDA #1  \ SET BUFF EMPTY
5290           STA bef \ FLAG
5300           RTS
5310
5320 .cnt4     PHP
5330           BCC cnt6
5340           LDA bff
5350           BNE cnt5
5360           JSR piltpo
5370           BEQ cnt5
5380           LDA bfl
5390           CLC
5400           ADC pol
5410           TAX
5420           LDA bfh
5430           ADC poh
5440           TAY
5450           SEC
5460           TXA
5470           SBC pil
5480           TAX
5490           TYA
5500           SBC pih
5510           TAY
5520           SEC
5530           TXA
5540           SBC bsl
5550           TAX
5560           TYA
5570           SBC bsh
5580           TAY
5590           PLP
5600           RTS
5610
5620 .cnt5     LDA pol
5630           SEC
5640           SBC pil
5650           TAX
5660           LDA poh
5670           SBC pih
5680           TAY
5690           PLP
5700           RTS
5710
5720 .cnt6     LDA bff
5730           BNE cnt7
5740           JSR piltpo
5750           BEQ cnt7
5760           LDA pil
5770           SEC
5780           SBC pol
5790           TAX
5800           LDA pih
5810           SBC poh
5820           TAY
5830           PLP
5840           RTS
5850
5860 .cnt7     LDA bfl
5870           CLC
5880           ADC pil
5890           TAX
5900           LDA bfh
5910           ADC pih
5920           TAY
5930           TXA
5940           SEC
5950           SBC pol
5960           TAX
5970           TYA
5980           SBC poh
5990           TAY
6000           TXA
6010           SEC
6020           SBC bsl
6030           TAX
6040           TYA
6050           SBC bsh
6060           TAY
6070           PLP
6080           RTS
6090 \
6100 \ SUBROUTINE WHICH RETURNS WITH
6110 \ THE EQUAL FLAG SET IF PI=PO
6120 \
6130 .pieqpo   LDA pil
6140           CMP pol
6150           BNE pie2
6160           LDA pih
6170           CMP poh
6180           BNE pie2
6190           LDA #0
6200           RTS
6210
6220 .pie2     LDA #1
6230           RTS
6240 \
6250 \ SUBROUTINE WHICH RETURNS WITH
6260 \ EQ FLAG SET IF PI<PO
6270 \
6280 .piltpo   LDA pih
6290           CMP poh
6300           BEQ pil2
6310           BCC pil3
6320           LDA #1
6330           RTS
6340
6350 .pil2     LDA pil
6360           CMP pol
6370           BCC pil3
6380           LDA #1
6390           RTS
6400
6410 .pil3     LDA #0
6420           RTS
6430 \
6440 \ SUBROUTINE TO PRINT A MESSAGE
6450 \ WHICH IS EMBEDDED IN THE CODE
6460 \ AND THEN CONTINUE AFTERWARDS
6470 \
6480 .print    PLA
6490           STA work
6500           PLA
6510           STA work1
6520           LDY #0
6530 .pm2      JSR incwork
6540 .pm3      LDA (work),Y
6550           BEQ pm4
6560           JSR osasci
6570           JMP pm2
6580
6590 .pm4      JSR incwork
6600           JMP (work)
6610 .incwork  INC work
6620           BNE inw2
6630           INC work1
6640 .inw2     RTS
6650 \
6660 \ SUBROUTINE TO GET THE X AND Y
6670 \ VALUES FROM THE STACK
```

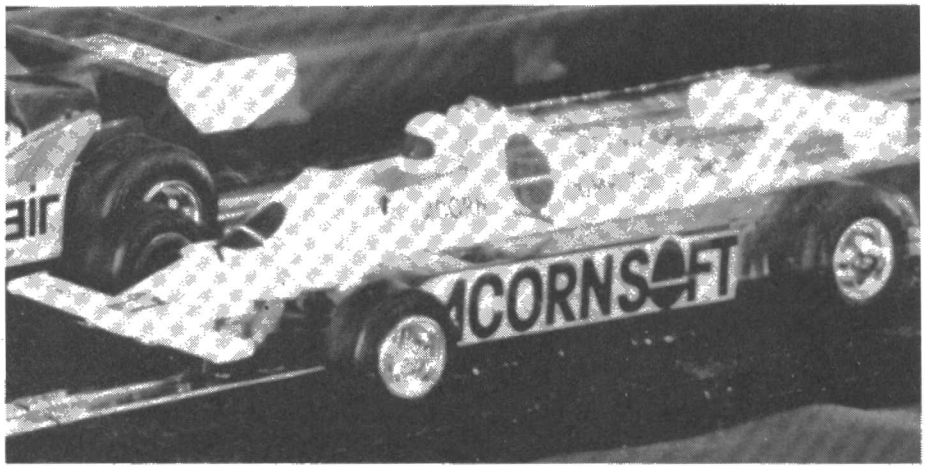
```
6680 \
6690 .getxy     TSX
6700            LDA &103,X
6710            TAY
6720            LDA &104,X
6730            TAX
6740            RTS
6750 \
6760 \ CODE TO ALLOW CONTROLLING SOFTWARE
6770 \ TO READ OR WRITE TO SIDEWAYS RAM
6780 \
6790 .substart CLC
6800            BCC ss2
6810            SEC
6820 .ss2       LDX &F4
6830 .ss3       LDY #&FF \ DUMMY RAM NUMBER
6840            STY &F4
6850            STY romselect
6860            BCC ss5
6870 .ss4       STA &FFFF \ DUMMY PI
6880            BCS ss6
6890 .ss5       LDA &FFFF \ DUMMY PO
6900 .ss6       STX &F4
6910            STX romselect
6920 .subend    RTS
6930 \
6940 \ CODE TO FIND SLOT NUMBER OF SIDEWAYS
6950 \ RAM IF PRESENT
6960 \
6970 .codeloc   LDA &F4
6980            PHA
6990            LDX #15
7000 .cd2       STX &F4
7010            LDA &8000
7020            STA bytes,X
7030            DEX
7040            BPL cd2
7050            LDX #15
7060 .cd3       STX &F4
7070            STX romselect
7080            LDA #0
7090            STA &8000
7100            LDA &8000
7110            BNE notram
7120            LDA #&FF
7130            STA &8000
7140            LDA &8000
7150            CMP #&FF
7160            BEQ ramfound
7170 .notram    DEX
7180            BPL cd3
7190 .cd4       PLA
7200            STA &F4
7210            STA romselect
7220            RTS
7230
7240 .ramfound  LDA bytes,X
7250            STA &8000
7260            CLC
7270            BCC cd4
7280 .codeend   NOP
7290 \
7300 \ CODE TO TEST FOR STAR COMMANDS
7310 \
7320 .comm      PHA
7330            TXA
7340            PHA
7350            TYA
7360            PHA
7370            LDX #&FF
7380            DEY
7390 .com2      INX
7400            INY
7410            LDA ctable,X
7420            BEQ cfound
7430 .com3      LDA (&F2),Y
7440            AND #&DF
7450            CMP # ASC(".")
7460            BEQ cfound
7470            CMP ctable,X
7480            BEQ com2
7490 .com4      INX
7500            LDA ctable,X
7510            BNE com4
7520            INX
7530            INX
7540            INX
7550            LDA ctable,X
7560            BEQ com5
7570            PLA
7580            PHA
7590            TAY
7600            JMP com3
7610
7620 \ COMMAND NOT FOUND SO LEAVE
7630 .com5      JMP end
7640 \
7650 \ COMMAND FOUND SO PUSH ACTION
7660 \ ADDRESS ON TO THE STACK AND DO
7670 \ AN RTI TO JUMP TO IT
7680 .cfound    DEX
7690 .cf2       INX
7700            LDA ctable,X
7710            BNE cf2
7720            INX
7730            LDA ctable,X
7740            TAY
7750            INX
7760            LDA ctable,X
7770            PHA
7780            TYA
7790            PHA
7800            PHP
7810            RTI
7820
7830 .ctable    EQUS "BUFFON"
7840            EQUB 0
7850            EQUW buffon
7860            EQUS "BUFFOFF"
7870            EQUB 0
7880            EQUW buffoff
7890            EQUS "PURGEON"
7900            EQUB 0
7910            EQUW purgeon
7920            EQUS "PURGEOFF"
7930            EQUB 0
7940            EQUW purgeoff
7950            EQUS "PURGE"
7960            EQUB 0
7970            EQUW purge
7980            EQUB 0
7990
8000 \
8010 \ CODE TO BE EXECUTED FOR *HELP
8020 \
8030 .help      PHA
8040            TXA
8050            PHA
8060            TYA
8070            PHA
8080            DEY
8090 .he2       INY
8100            LDA (&F2),Y
8110            CMP # ASC(" ")
8120            BEQ he2
8130            CMP #13
8140            BNE he5
8150            LDX #&FF
8160 .he3       INX
8170            LDA helpmess,X
8180            BEQ he4
8190            JSR osasci
8200            JMP he3
8210
8220 .he4       JMP end
8230
8240 .he5       LDX #&FF
8250            DEY
8260 .hloop     INX
8270            INY
8280            LDA htable,X
8290            BEQ hfound
8300 .hlo2      LDA (&F2),Y
8310            AND #&DF
8320            CMP # ASC(".")
8330            BEQ hfound
8340            CMP htable,X
8350            BEQ hloop
8360 .hlo3      INX
8370            LDA htable,X
8380            BNE hlo3
8390            INX
8400            INX
8410            INY
8420            LDA htable,X
8430            BNE hlo4
8440            JMP end
8450
8460 .hlo4      PLA
8470            PHA
8480            TAY
8490            JMP hlo2
8500
8510 .hfound    DEX
8520 .hf2       INX
8530            LDA htable,X
8540            BNE hf2
8550            INX
8560            LDA htable,X
8570            STA work
8580            INX
8590            LDA htable,X
8600            STA work1
8610            LDA #13
8620            JSR osasci
8630            LDY #&FF
8640 .hf3       INY
8650            LDA (work),Y
8660            BEQ hf5
8670            JSR osasci
8680            CPY #&FF
8690            BNE hf3
8700            INC work1
8710 .hf4       JMP hf3
8720
8730 .hf5       LDA #13
8740            JSR osasci
8750            JMP endz
8760
8770 .htable    EQUS "BUFFON"
8780            EQUB 0
8790            EQUW hbuffon
8800            EQUS "BUFFOFF"
8810            EQUB 0
8820            EQUW hbuffoff
8830            EQUS "PURGEON"
8840            EQUB 0
8850            EQUW hpurgeon
8860            EQUS "PURGEOFF"
8870            EQUB 0
8880            EQUW hpurgoff
8890            EQUS "PURGE"
8900            EQUB 0
8910            EQUW hpurge
8920            EQUB 0
8930
8940 .helpmess EQUB 13
8950            EQUS name$+" "+version$
8960            EQUB 13
8970            EQUS "  BUFFON"
8980            EQUB 13
8990            EQUS "  BUFFOFF"
9000            EQUB 13
9010            EQUS "  PURGEON"
9020            EQUB 13
9030            EQUS "  PURGEOFF"
9040            EQUB 13
9050            EQUS "  PURGE"
9060            EQUB 13
9070            EQUB 0
9080 .hbuffon  EQUS "Enable sideways RAM pr
inter buffer.":EQUB 13
9090            EQUS "An error will be repor
ted if the":EQUB 13
9100            EQUS "current language is in
sideways RAM.":EQUB 13
9110            EQUS "The command will remai
n enabled after":EQUB 13
9120            EQUS "a CTRL/BREAK, although
the contents of":EQUB 13
9130            EQUS "the buffer will be los
t.":EQUB 13
9140            EQUB 0
9150
9160 .hbuffoff EQUS "Disable sideways RAM b
uffer. You":EQUB 13
9170            EQUS "cannot switch the buff
er off until":EQUB 13
9180            EQUS "it is empty.":EQUB 13
9190            EQUB 0
9200
9210 .hpurgeon EQUS "This command enables t
he ESCAPE key to":EQUB 13
9220            EQUS "purge the sideways RAM
printer buffer.":EQUB 13
9230            EQUB 0
9240
9250 .hpurgoff EQUS "This command prevents
the ESCAPE key":EQUB 13
9260            EQUS "from purging the sidew
ays RAM printer":EQUB 13
9270            EQUS "buffer. This is the de
fault condition.":EQUB 13
9280            EQUB 0
9290
9300 .hpurge   EQUS "This command will purg
e the sideways":EQUB 13
9310            EQUS "RAM printer buffer.":E
QUB 13
9320            EQUB 0
9330 .romend    NOP
9340 ]
9350 pl=&880+ss4-substart+1
9360 ph=&880+ss4-substart+2
9370 pol=&880+ss5-substart+1
9380 poh=&880+ss5-substart+2
9390 rno=&880+ss3-substart+1
9400 bytes=&100+codeend-codeloc
9410 NEXT
9420
9430 REM CALCULATE CHECKSUMS FOR
9440 REM 128 BYTE BLOCKS
9450
9460 DATA 10063,13819,15444,14410,14527
9470 DATA 13492,13005,13884,12573,13435
9480 DATA 14648,17190,15516,19513, 8900
9490 DATA 11599,11305,11316,11621
9500 PRINT'"STARTING CHECKSUMS"'
9510 E%=0
9520 FOR I%=code% TO O%-128 STEP 128
9530 S%=0
9540 FOR J%=0 TO 127
9550 S%=S%+(I%?J%)
9560 NEXT
9570 READ T%
9580 IF S%<>T% THEN PRINT"ERROR IN BLOCK FR
OM &";~I%+&8000-code%;" TO &";~I%+&807F-code
%:E%=1
9590 NEXT
9600 IF E%=1 THEN PRINT'"MAKE SURE THE CHEC
KSUM NUMBER IN THE"'"DATA STATEMENT IS CORRE
CT BEFORE"'"CHECKING THE CODE"':END
9610 PRINT"CHECKSUMS OK"'"SAVING CODE"
9620 OSCLI("*SAVE OBJCODE "+STR$~(code%)+"
+A00 8000 8000")
```

**Please Note** – The authors are prepared to supply copies of the program on tape, 40 or 80 track disc for £8 including p&p. Please state which you require. Readers with BASIC 1 will be provided with a machine code file ready for use in sideways RAM or for programming into an EPROM if this is requested when placing their order. The authors will also program EPROMs which are sent to them with the machine code output from the listing for a similar amount. Please send orders to:

*Mr. B. V. Alderwick,*
*40 The Chase,*
*Cashe's Green,*
*STROUD,*
*Gloucester*
*GL5 4SB*

# KEEPING TRACK OF SLOT CAR RACERS

## Accurate timing of slot car events is possible with Paul Beverley's latest interface for the BBC micro.




If you are a beginner in the world of interfacing, here is a simple and yet fascinating project for you to try. It is ideal for a novice because it involves only a very small amount of electronics hardware and uses one of the many interfaces available on the BBC microcomputer. The aim of the project is to time your slot-car – or even your model train.

In this article we will consider the hardware needed and look at a simple BASIC program which will enable you to measure instantaneous speeds and lap times and to calculate the average speed of a single car or train travelling round a fixed track. Further extensions to the hardware and the software will enable us to measure acceleration. The timing will be done using the computer's centisecond clock which will give the time to the nearest one-hundredth of a second. Later we will look at ways of timing two cars at the same time, and of improving the resolution of the timing and allowing a continuous readout of the speeds and lap timings.

## The Hardware

The interface used is the A/D converter input – the 15-way connector on the back of the BBC microcomputer that is used for the joysticks. This interface can be used to take in a voltage (somewhere between zero and 1.8 volts) and will present the user with a number (between 0 and 65520) which is proportional to the voltage applied.

The only external hardware that is needed is an infra-red detector, which will cost you around 80p, and a single resistor. The circuit diagram, if it can be called as such, is given in **Figure 1.** The resistor is used to supply current to the detector from the reference voltage coming out of the ADC connector. The idea is that if the detector is not illuminated then it will not conduct at all, and the full 1.8 volts reference voltage is fed into the analogue input. As the illumination on the detector increases, the current through it, and hence through the resistor, increases so that the voltage at the ADC input decreases. At a certain illumination level there will be enough current to ensure that the voltage across the detector is so small that the ADC registers zero volts.

The value of the ADC input voltage will be given by:

$$V_{adc} = V_{ref} - I_d * R$$

The current which the detector can draw ($I_d$) depends solely on the intensity of the infra-red illumination and is not affected by the value of the resistor. Therefore increasing the value of the resistor will mean that the voltage at the ADC input will reach zero at a lower current and thus at a lower illumination level, making the system more sensitive. Conversely a smaller resistor will mean that the illumination will have to be higher before zero volts is reached.

We are not actually using the detector to measure the illumination but simply to detect when the car moves over the top and obscures it. Therefore we are using it in an ON-OFF or digital mode. It may seem strange, therefore, to be using an analogue input rather than a digital input such as the user port. However the user port is nowhere near as sensitive as the analogue input and we would need extra hardware to provide the necessary sensitivity. The analogue input is extremely sensitive as it has an input resistance of several million ohms.

The only problem with using the

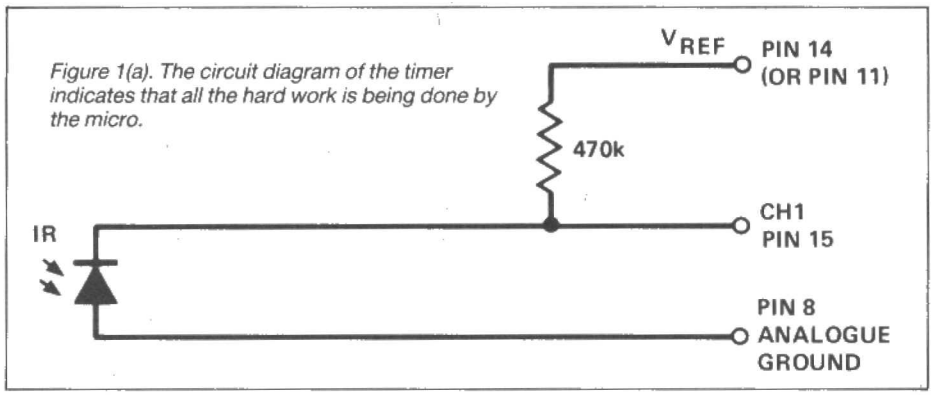



*Figure 1(a). The circuit diagram of the timer indicates that all the hard work is being done by the micro.*

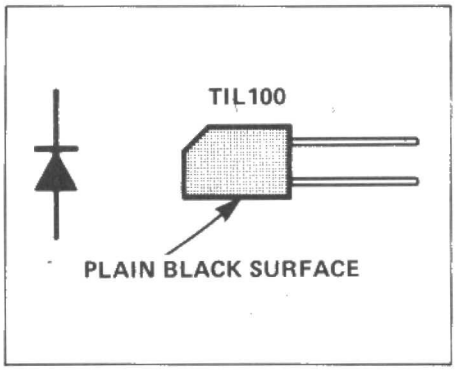



*Figure 1(b). Detail of the IR LED used.*

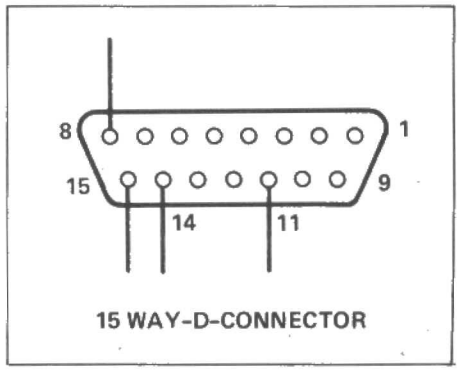



*Figure 1(c). Pin assignments of the BBC's A/D connector.*

**LISTING 1. The lap-timer software.**

```
 10 REM Single track lap-timer
 20 REM ***********************
 30 REM (C) Norwich Computer Services
 40 MODE3
 50 VDU19;4;0;
 60 record=0:REM or current lap record in MPH
 70 track_length_cm=4*(34.9+25.6+32.3)+2*35.8
 80 car_length_cm = 13
 90 *FX16,1
100 REPEAT
110   INPUT"Number of laps";laps%
120   IF laps%<1 THEN @%=10:END
130   record_run=FALSE
140   previous_T%=0
150   FOR N%=1 TO 3
160     WAIT=INKEY100
170     VDU7
180     NEXT
190   VDU7
200   TIME=0
210   WAIT=INKEY200
220   FOR lap%=1TOlaps%
230     REPEATUNTILADVAL1
240     T%=TIME
250     REPEATUNTILADVAL1=0
260     U%=TIME
270     IF U%=T% PRINT '"Cheat":U%=T%+1
280     @%=&90A
290     PRINT"Lap ";lap%;
300     @%=3
310     PRINT TAB(10)"Inst. speed =";
320     PRINT INT(car_length_cm/(U%-T%)*72);
330     @%=&20103
340     lap_speed=track_length_cm/(T%-previous_T%)*72
350     PRINT TAB(33)"Lap speed =",lap_speed;
360     PRINT TAB(56)"Average speed =";
370     PRINT ,track_length_cm*lap%/T%*72
380     previous_T%=T%
390     IF lap_speed>record record=lap_speed:record_run=TRUE
400     NEXT
410   IF record_run PRINT '"New lap record = ";record " MPH"
420   PRINTCHR$(7)
430   UNTIL0
440 END
```

analogue input is the time which it takes to register a change of input voltage. If you are using only one of the four channels, then the ADC value is checked once every 10 milliseconds. This means that we cannot do any better than measure the timing of the car to the nearest centisecond. Therefore it is quite acceptable to use the TIME function in BBC BASIC, which gives a measure of time in centiseconds. This gives quite an acceptable accuracy for the lap timings since we are measuring times of a couple of seconds or more to the nearest hundredth of a second.

When it comes to measuring instantaneous speed, however, the accuracy is nowhere near as good. When the cars are moving at full speed, the time taken to pass over the detector can be as little as 5 or 6 centiseconds which puts a severe limit on the accuracy. However it is still worth doing as it does give an indication of the speed even if we cannot say that it is an accurate measurement.

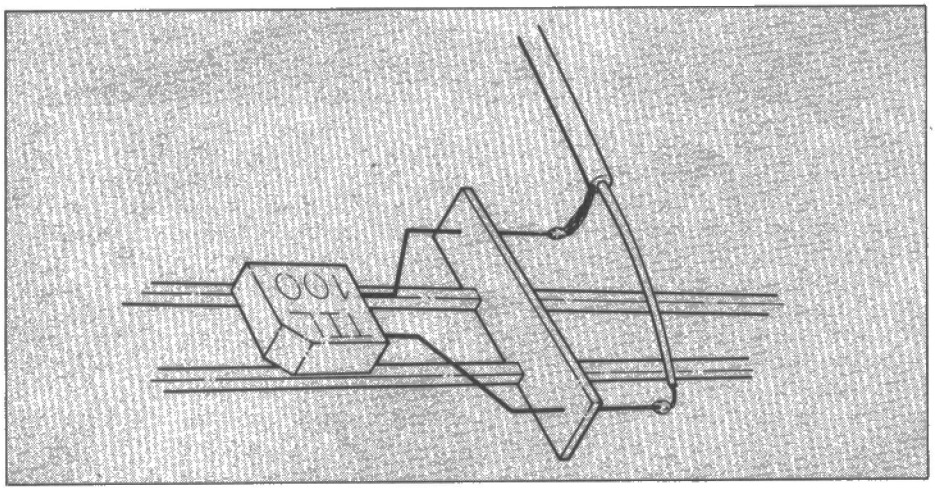


*Figure 2. The IR LED may be secured directly under one of the track's slots.*

It is in fact possible to speed up the conversion process on the ADC so that it gives a value every four milliseconds. There would be no advantage in doing so, however, as long as we were using the TIME function since this only measures to the nearest 10 milliseconds. To measure more accurately we would have to use the timers in the 6522 VIA. This is more complicated and would require the use of assembly language programming.

The infra-red detector itself is a rectangular block of plastic with two wire connections which can easily be secured directly underneath one of the slots on the track.

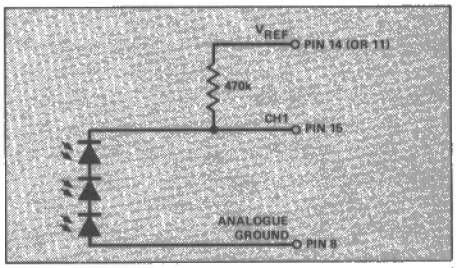



*Figure 3. Series LEDs allow acceleration measurements.*

The simplest way to do this is to bend the legs so that they can be pushed through one of the plastic cross-ribs which run on the under-side of the track. This is shown in **Figure 2.** The wires can be pushed through the plastic by heating them up a little bit with a soldering iron while holding them with a pair of pliers. The resistor can be mounted inside the 15-way plug, being joined to the infra-red detector by a piece of co-axial cable. The outer conductor is used as the ground connection. You can easily use several metres of cable without any difficulty.

Although we are using an infra-red detector, we do not need to use an infra-red source as such. An ordinary filament lamp can be used as these give a lot of infra-red radiation as well as visible light. A 60-watt lamp placed 3 or 4 feet about the track, is quite sufficient to operate the detector when using a 470k resistor.

## Software

If we use X% = ADVAL(1) (or ADVAL1 since the brackets are optional) we get a number which relates to the voltage appearing at the first analogue input. The value returned will tell us whether the detector is obscured or not. When the detector is fully illuminated, the value it gives is zero. If you then obscure the detector with your hand, you will see that the value shoots up to several thousand. All we have to do therefore is to check whether or not the ADVAL value is actually equal to zero, and use this to detect when the car moves over the detector.

The program given in **Listing 1** first of all provides a measure of the instantaneous

**LISTING 2. Acceleration measurement software.**

```
  10 REM Acceleration Measurement
  20 REM of Slot Cars.
  30 REM (C) Norwich Computer Services
  40 s=34.9:REM sensor separation in cm
  50 REPEAT
  60   REPEATUNTILADVAL1:TIME=0
  70   REPEATUNTILADVAL1=0
  80   REPEATUNTILADVAL1:T%=TIME
  90   REPEATUNTILADVAL1=0
 100   REPEATUNTILADVAL1:U%=TIME
 120   U%=U%-T%
 130   a=2*s*(T%-U%)/(T%*U%*(T%+U%))*1E6
 140   PRINT"Acceleration = ";INT(a);" cm/s/s"
 150   UNTIL0
 160 END
```

speed of the car as it passes over the detector. It does this by measuring the length of time for which the detector is obscured and then doing a calculation using the length of the car. The lap time is simply measured by taking the difference between the time at which the car begins to obscure the detector on the first occasion, and the time at which it obscures it again. The average speed for each lap can also be calculated if the length of the track is known. The overall length of the track can be calculated by use of one of the track length indicators available from the toy shops which sell the cars.

Since we are only timing one car, it is very easy for the computer to spend a little time calculating and displaying the results immediately after the car has gone over the detector and yet still be ready by the time the car comes round again at the end of the lap. If we want to go on to deal with two cars at the same time the problem is more complicated since it is possible for the seond car to go over its detector while we are dealing with the fact that the first car has done the same. Therefore a slightly different approach is required. This willl be dealt with in next month's article.

## The program

Having set up an appropriate mode with a blue background at lines 40 and 50, the value of the latest lap record is set up at line 60. If the lap record is broken during the course of running of the program, you can change the value given, and save the program again for later use. At line 70 the track length is calculated from the number of pieces of track of each given length, whether they are straight, curved, or crossover. This length is given in centimetres, as is the car length at line 80. The FX call at line 90 selects only one analogue converter channel to speed up the conversion. The loop which then follows allows you to do any number of runs, each of a specified number of laps. The number of laps is entered by the user and the variables used in the run are initialised.

Lines 150 to 190 are used to produce a start sound. This consists of three bleeps, the third of which is the longest. The idea is that the car is allowed to start as soon as the third bleep begins. At this point the TIME value is set to zero. It then delays for two seconds (line 210) before the first time at which it checks whether the cell has been obscured. The reason for this is that the car must start with its front bumper just behind the detector, then one complete lap will bring it round so that as the bumper obscures the detector, it indicates that the lap has been completed.

For the number of laps selected (line 220) the program waits until the ADVAL1 value is greater than zero and then it records the length of time since the beginning. It then waits until ADVAL1 is equal to zero again; is when the car has gone completely over the detector, and records the time again. Line 270 is used to stop people cutting the beam with their hand and getting a false reading. The number of the lap and the instantaneous speed are then printed out at lines 280 to 320, and then the lap speed is calculated, as is the overall average speed at that stage in the race.

If the lap speed is greater than the lap record (line 390) the new value is recorded, and the fact that a record has been made is registered. At the end of the race, if a lap record has been created, the fact is displayed on the printout. All the speeds that are printed out are given as scale speed in miles per hour. These are calculated by using the conversion factor given with the slot car information.

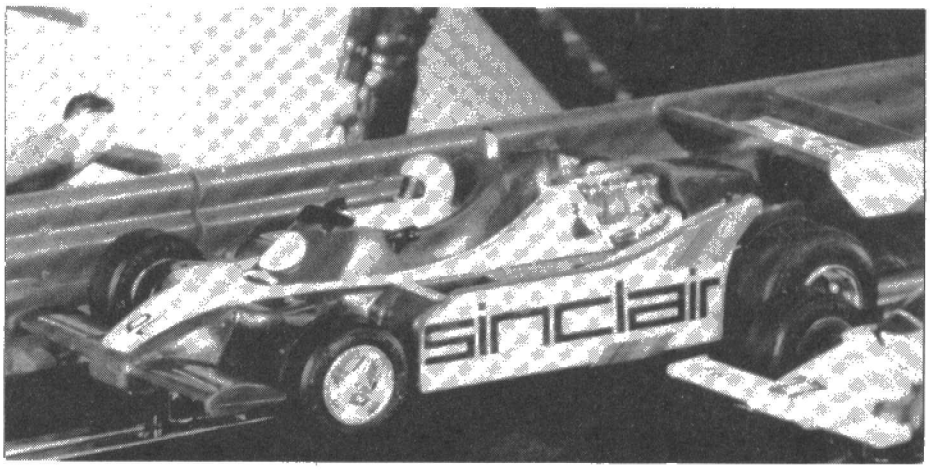
## Measuring acceleration

These infra-red detectors are so sensitive that it is possible to put two or three of them in series **(Figure 3).** Then as long as all three are illuminated, the value given by ADVAL will be zero. If three such detectors are used and placed at equal intervals along the track, then the acceleration of a car can be measured. **Listing 2** shows a program which will do this. The detectors were placed one on each of three successive pieces of track, each of which was 34.9 cm long. The program simply records the times taken from the first detector to the second, and from the second to the third.

The calculation at line 130 assumes that the acceleration is constant, which may not of course be true, but it does give a good indication of the state of the car. If its brushes are badly worn, you will find that the acceleration is noticably reduced. Renewing the brushes can improve the acceleration quite considerably. You can also measure the difference in acceleration between different cars, and even examine the effect of cleaning the wheels and generally servicing the cars.

## Applications

This project is quite fascinating in itself, but the principles used can be applied in other circumstances. Even this simple hardware can be applied to both the practical alterations. The use of infra-red detectors attached to the analogue port of the BBC microcomputer could be applied to detecting objects passing down a conveyor belt for example. They could even be used to time real cars or go-karts, as long as a suitable illumination was provided. This has been tried using these infra-red detectors for timing go-karts, and it was found that a car head-light was quite sufficient illumination even in normal daylight. All that was necessary was to place the detector at one end of a cardboard tube, the inside of which has been blackened. This stopped reflection from outside light operating the detector. If you think up other suggestions for applications, let us know.



# A BOX OF TRICKS

## The Microbox II is a 6809 based single computer that packs in features that, in the opinion of Mike James, make it a remarkable machine.

At first sight the Microbox II looks like another of the many single board 6809 machines that have started to appear on the market – it certainly is no more expensive – yet on closer examination it is packed with features that make it a remarkable machine at any price. By using a combination of 'RAM disc', 'EPROM disc' and a traditional five inch floppy it offers a flexibility of data storage and convenience that is difficult to beat without using an expensive Winchester disc.

By providing both an advanced memory mapped graphics controller (the NEC7220), a parallel keyboard port and a pair of RS232 interfaces, the system can be configured to work with monitor and keyboard or traditional VDU or a mixture of the two. By supplying a real time clock/calendar combined with an area of battery backed up RAM, not only is the correct time and date always available but any configuration changes can be stored while the power is switched off.

Many of the other good points of the Microbox II only become apparent when you see how they fit into the overall design or how the supplied software makes use of them. Specifically if you are interested in a small 6809 machine running the Flex operating system with a very fast response time for loading data or command files, or if you are in the least bit interested in high resolution graphics at speed, then you need to consider the Microbox II.

### Kits come back?

Back in the early days of microcomputing the only real way to acquire a machine was to build it yourself. Some of the less intrepid enthusiasts took the easier option of putting a kit together – a lot of people learned a substantial amount about computer hardware by this route! The reason for this sudden outburst of nostalgia is that Microbox II is available in the form of a printed circuit board and a set of constructional notes! I have to say at this stage that the Microbox II that I received for review was already built but after examining the layout and construction notes I would have no hesitation in trying to put one together. The prerequisites are an ability to solder reasonably well and some idea of what each component is and how carefully it should be treated. In other words, most readers of *E&CM* would have no trouble with the project! It is worth noticing that while Micro Concepts offer a package consisting of the PCB, 8K ROM monitor and Flex support disc as the lowest cost way of acquiring a Microbox II, they also supply complete systems to order. Making a machine available as a bare PCB or as a kit makes very good sense when you are offering an advanced design that, irrespective of its merits, is unlikely to rank among the biggest sellers of all time. To complete the construction you will also need a +5V, +12V and –12V power supply, either a keyboard and a video monitor (notice that a TV set will not do) or a standard RS232 VDU.

### The CPU and graphics

The CPU section of the Microbox II is very simple because of its use of up to date chips. A 6809C CPU is interfaced to 64K of dynamic RAM (eight 6164 chips) by an MC6883 otherwise known to Dragon and Tandy Color Computer owners as a SAM (Synchronous Address Multiplexer) chip! This remarkable chip replaces a board full of conventional logic and performs memory refresh, clock generation (for the 6809), row and column address multiplexing and looks after much of the address decoding. A single 2764 EPROM provides the storage necessary for the 8K of monitor software used by MicroboxII. This combination provides in eleven chips the largest hardware environment that Flex can use! The review model was a standard single speed 1MHz model but Micro Concepts have a double speed Microbox available.

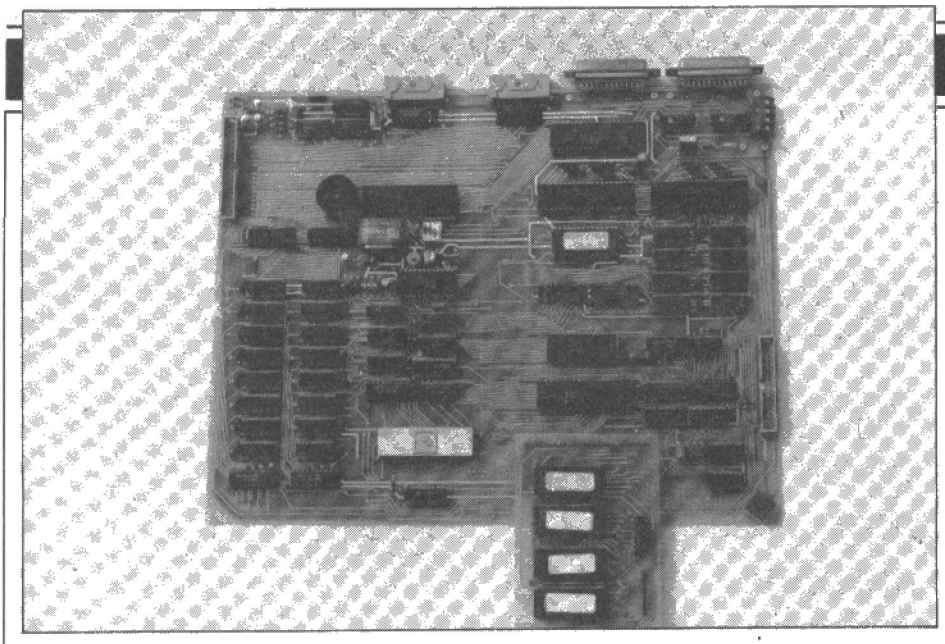
Moving away from the 6809, the next largest section is the graphics and RAM disc circuit. Graphics are provided by a single innovatory LSI chip, the NEC/220, and a handful of standard TTL (LS) chips. The NEC7220 is a graphics controller chip which is fairly remarkable. Most graphics chips content themselves with generating a video signal by reading data from an area of shared RAM not so the NEC7220 – it uses its own dedicated RAM, 128K of it in the case of the Microbox! At first it looks as though video controller design has taken a step backward in that not sharing the video RAM between the controller and the CPU means that somehow vast quantities of data have to be moved between the CPU's RAM and the dedicated video RAM. You could say that the whole basis of fast low-cost graphics is the sharing of RAM between the CPU and a display chip – the CPU is responsible for depositing the data and the display chip is responsible for displaying it – and the pair get on with their job with as little interaction as possible. However this neat picture was before the NEC7220 appeared! By making a graphics controller that is as complicated, if not more so than a state of the art microprocessor, NEC have managed to remove almost all of the responsibility for generating and manipulating graphics data from the CPU. The CPU can give short commands to the NEC7220 that cause it to draw lines, circles or fill areas.

This high level command structure means that there is absolutely no need for the CPU to involve itself in directly altering the contents of the video RAM and hence there is no need for the CPU to share the video RAM with the controller. An immediate consequence of this separation of system RAM into two parts is that the size of the video RAM isn't limited to whatever the CPU can reasonably address. Thus while the 6809 can only directly address 64K of memory the Microbox has an additional 128K of RAM dedicated to graphics applications. This extra 128K is supplied in the form of an additional sixteen 6164 dynamic RAM chips which are addressed, refreshed etc by the graphics controller – that is they form an entirely separate memory system from the 64K addressed by the 6809.

What the use of the NEC7220 and 128K of video RAM means in terms of features is difficult to appreciate without a demonstration. In text mode the Microbox can display up to 128 columns x 72 lines, that is more than an A4 page, but you would need a special monitor to make use of such a display. More usually the text mode would be used at 108 columns x 24 lines or 84 columns x 24 lines, again with a video monitor. In high resolution graphics mode the resolution is a staggering 768 x 576 pixels – that is around 400 thousand pixels in total! (This should be compared to the BBC Micro's highest resolution mode offering 640 x 256 pixels.) What is most surprising about the high resolution graphics offered by the Microbox is the combination of resolution and high speed. Other features include a lightpen input, hardware scrolling, hardware pan and zoom, etc. For complete details you will have to consult the NEC7220's data sheet! From the programmer's point of view there is no need to get involved in the hardware however as a complete set of assembly language macros are supplied for a wide range of graphic operations from plotting a point, to lines, rectangles and circles.

### RAM and EPROM disc

Not all of the 128K of video RAM is in fact used for display purposes – the remainder



provides the storage for a RAM disc. Most designers would have simply allowed this leftover memory to lie idle – but not the designer of the Microbox II. A little software converts it into something that is usually considered an extra on other systems.

If you have never come across the idea of a RAM disc then it might seem a little strange at first but its usefulness is undeniable. An operating system such as Flex creates named files out of chunks of storage called 'sectors'. Normally these sectors are stored on some form of magnetic media – ie a floppy, a three inch or a Winchester disc. However there is nothing that says a sector shouldn't be stored in an area of RAM for example. This is indeed what happens in a RAM or 'semi' disc, an area of memory is organised into sectors which the operating system treats in exactly the same way as sectors on a magnetic disc. For example you can write and read named data files to and from Microbox's 128K of video RAM just as if it was an additional disc drive – however there are two important differences.

Firstly, as RAM has no moving parts, RAM disc is very fast, roughly ten times faster than a standard 5″ disc. For small files this makes access times virtually instantaneous. Secondly RAM is volatile, if you switch the Microbox off then all your RAM files will be lost. As you can imagine this volatility makes RAM disc less useful than magnetic disc but there are many applications where termporary files are created during processing and in such cases RAM disc would speed things up no end. In practice a RAM disc is best seen as a storage device to be used along with a standard magnetic disc to get the best performance out of a system. For example if you wanted to sort a file then it will almost certainly be worth copying to RAM disc, sorting it and then copying it back to magnetic disc! The amount of RAM disc storage available depends on what graphics features are in use. If the video is not in use, ie you are using the RS232 interface and a VDU to communicate with Microbox, there are 500 sectors, if you use the video for text display this is reduced to 390 sectors and if you are using the high resolution graphics this comes down to 180 sectors. As a sector is 256 bytes of storage you will realise that unless you are using high resolution graphics the RAM disc is roughly equivalent to a 40 track 5″ disc in storage capacity.

The provision of such a potentially large amount of RAM disc is interesting enough but Microbox II goes one step further than this. In the same way that named files can be stored in RAM they can be stored in read only memory. Obviously such files could only be read back into memory but this is far more useful than you might imagine. For example consider what happens when you type a simple command such as CAT (catalogue a disc) to a Flex system. Before the system can respond with a catalogue as requested the utility that does the cataloging has to read in – that is CAT causes a program called 'CAT' to be loaded and executed. So it is with nearly all Flex commands and this loading introduces a delay into doing almost anything. In some cases the delay can be tens of seconds, for example when a text editor is requested. Now think of the difference to the overall performance of the system if these command utilities and application programs were stored in a read only version of RAM disc – almost instantaneous response to any command line! This is acactly what you get if you make use of Microbox's 256K of EPROM disc. From the hardware point of view all this consists of is four EPROMs interfaced to the system by an 8255 PPI (Intel's equivalent of the well known 6820 PIA or the 6522 VIA) but from the user's point of view it is like magic! You can load all of your commonly used system software into EPROM (subject to it fitting into 1024 sectors) and from then on you can have response times that previously could only be achieved by using expensive Winchester discs. Finally the really pleasing point is that by applying 21V to connection on the EPROM board you can use Microbox II to program its own EPROMs – even the software is included!

## The rest of the system

After all this talk of RAM discs and EPROM discs it hardly seems worth commenting that there is a standard single/double density WD1770 chip that will look after two 40/80 single or double sided disc drives! Also by comparison the WD2123 dual RS232 interface seems quite dull but if you want to use either a serial printer or a VDU then it is essential. Even the MC146818 clock/calendar, a chip that I would have spent a lot of time describing had it turned up in a less interesting machine seems mundane but it has been integrated into the system very nicely. As well as providing the time and date its Nicad battery backup enables its scratch pad RAM to be used to store startup parameters making software configuration of the system permanent. The final large chip in the system is a common 6821 used for system configuration via dip switches, a beeper, a printer interface and a parallel keyboard input port. All of these features add to the Microbox's versatility and you cannot help but get the impression that the designer has thought about how the system might be used.

## Using Microbox II

One of the problems with selling a bare PCB is that it gives the potential user very little idea of what the final machine might look like. In the case of the Microbox II this if further complicated by the fact that you can quite happily contemplate building a machine with no disc drives! More realistically this is the first machine that can conveniently be used with one 5″ floppy drive! With all of the system software in EPROM disc and temporary files in RAM disc this allows a single floppy to be dedicated to its natural task of data storage. Even the notorious single disc copying problem (almost) vanishes because you can copy a 5″ disc into RAM disc and then back to a different 5″ disc. If you think about it this, coupled with the speed of EPROM and RAM disc and the high quality of Flex system and application software makes Microbox II an ideal low cost business system. At the moment my ideal word processor would be a single drive Microbox II with a detachable keyboard and swivel mounted monitor – if only I could find the right case for it at the right price. On the other hand the very high resolution monochrome graphics makes it suitable for technical applications, data logging, 3D graphics etc. There is even an expansion bus that provides all the signals needed to interface odd peripherals with very little effort.

In some ways Microbox II makes me a little sad in that it is unlikely ever to reach a mass market of the sort that it deserves because it is a very attractive proposition that combines the forefront of technology with a remarkable degree of economy.

Microbox II is available from Micro concepts, 8 Skillicorne Mews, Queens Road, Cheltenham.

The Microbox II Startup Kit (the bare PCB plus the 2764 EPROM) costs £95.

Built and tested units are available at £595 plus VAT and cased units complete with disc drives cost from £900 plus VAT.

# STRAIGHTENING THE CIRCLE

## The Spectrum's circle command is oh so S-L-O-W. JR Hibberbine's program speeds things up by forming circles from a series of straight lines.

Like most other home computers the Spectrum comes equipped with a command for drawing circles with the minimum of fuss. Unfortunately, the CIRCLE command in the Spectrum is rather s-l-o-w. Type in CIRCLE 128,87,80 and you might as well fix yourself a sandwich while you wait for the result to appear. A little unfair? Perhaps, but it can't be denied that an increase in speed would be welcome. The question is, can we do better? Most certainly we can!

The Spectrum circle generator works by drawing a series of short straight lines which approximate the path taken by a 'true' circle. In detail the procedure is complicated, but it's essential characteristics are shown by the BASIC program in **Listing 1** (the comments in the listing should clarify everything except for the maths!).

Now, without going into too much detail, we can determine the reasons why this method is so slow. Firstly, because of its reliance on the Sine and Cosine functions, the routine has to use real (floating point) arithmetic. To calculate each set of DRAW parameters (DX and DY) four floating-point multiplications and two floating-point additions are required (lines 90 and 100 in **Listing 1**).

The problem here is that the Spectrum's floating-point calculator was really designed to be compact and accurate at the expense of being fast. As a result it makes heavy going of all this work.

Secondly, the number of times the routine has to go through the above calculations depends on the value of the angle A (in **Listing 1** A is given an arbitrary value of 10 degrees, whereas in the Spectrum ROM the value is worked out such that the larger the radius, the smaller the angle). In particular, the real arithmetic calculations of lines 90 and 100 will be executed 360/A times. This is actually quite unnecessary.

Take a look at **Figure 1,** which shows a circle divided up into eight equal slices (S1 to S8), and note that in each slice an initial straight line segment is drawn (LS1 to LS8). Given that the parameters for LS1 are DX and DY, then the parameters for LS2 to LS8 can be obtained simply by changing the signs and/or swapping around the values of DX and DY as shown in the diagram. This means that only the Sine and Cosine equations are needed to calculate DX and DY for one slice (ie 1/8th) of the circle.

It should be obvious from the above points that the ideal circle generator is one which takes advantage of the symmetry of the circle and does not use real arithmetic. Before going on to consider such a routine, bear in mind a point in favour of the method in **Listing 1;** that is, it is quite easy to adapt the program so that it can generate other shapes as well. As an example, try introducing a new variable, E, into the program with the line

```
25  INPUT "E ";E
```

and then alter line 70 as follows:

```
70  DRAW DX,E*DY
```

As long as E=1 then circles will be drawn. Any other value (say between 0 and 5) will produce elipses.

As we have seen, the Spectrum draws circles by calculating the parameters for a series of short straight lines and then passing these parameters to the DRAW routine. A more direct approach is to have the circle generator itself produce all the appropriate pixel co-ordinates. These would then be passed to the PLOT routine. Rather than actually calculate the co-ordinates of each individual point that lies on the circle (as is the case in the example given on page 125 of the Spectrum man-

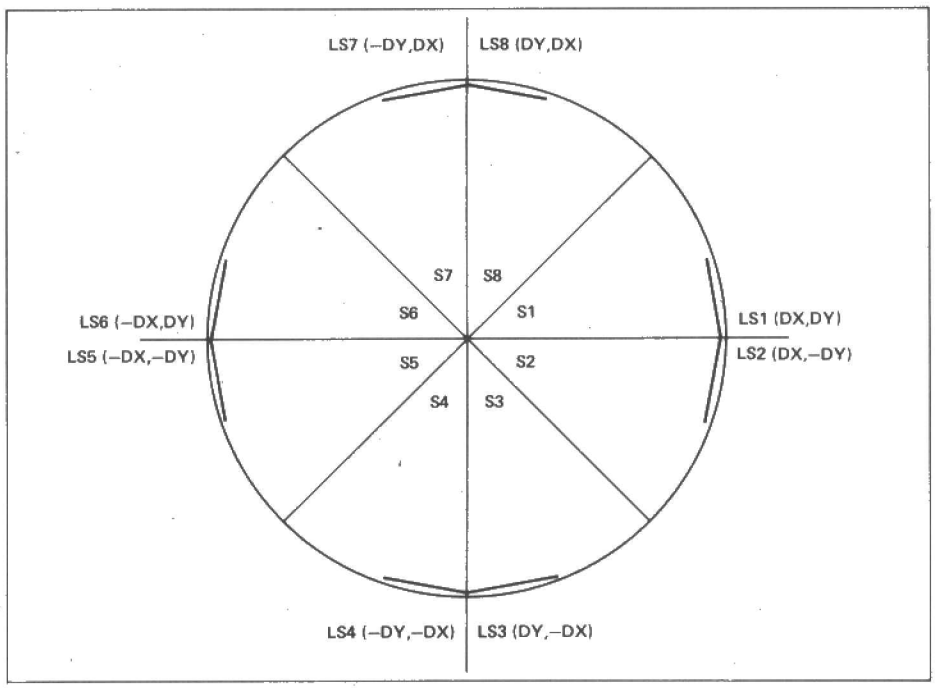



*Figure 1. Only the sine and cosine equations are required to calculate DX and DY.*

**LISTING 1.**

```
10 INPUT "CIRCLE ";CX;" ";CY;" ";R:  REM    Usual CIRCLE parameters.
15 REM
20 LET A=10: LET AR=A/180*PI:  REM          The angle A determines size
25 REM                                      of straight lines segments.
27 REM
30 LET C=360/A:  REM                        C is no. of line segments
32 REM                                      to be drawn.
33 REM
35 LET SINEA=SIN AR: LET COSINEA=COS AR
40 LET DX=0: LET DY=R*SIN AR:  REM          1st set of DRAW parameters.
45 REM
50 PLOT CX+R,CY-DY/2:  REM                  Circle starting point,
55 REM                                      just above 3 o'clock position.
60 REM
70 DRAW -DX,DY:  REM                        DRAW a line segment.
72 REM
75 LET C=C-1: IF C=0 THEN  STOP :  REM      Program stops when C line
80 REM                                      segments have been drawn.
82 REM
85 LET TEMPDX=DX:  REM                      DRAW parameters
90 LET DX=DY*SINEA+DX*COSINEA:  REM         for next line segment
100 LET DY=DY*COSINEA-TEMPDX*SINEA: REM     are calculated.
105 REM
110 GO TO 70:  REM                          Go back to DRAW next line.
```

ual), it is far more efficient to use what is known as the *incremental* method. That is, having plotted one point (X,Y) the next point is derived simply by incrementing (or decrementing) either one or both of the co-ordinates.

For example, consider **Figure 2a** which shows a one-eighth slice of a circle with centre co-ordinates (0,0). We start by plotting the point at the 3 o'clock position (X1,Y1) where X1 is equal to the radius and Y1=0. **Figure 2b** shows pixel positions (somewhat oversized!) superimposed on the path of the curve. As you can see, the shaded pixels are the ones that follow this path the closest.

If you look at this figure carefully, you will see that, for one of the shaded pixel points (call is X1,Y1) which lie on the curve, the next relevant point (X2,Y2) is derived in one of the following ways:

| | | |
|---|---|---|
| EITHER | X2 = X1 | : no change |
| | Y2 = Y1+1 | : increment Y co-ordinate |
| OR | Y2 = X1–1 | : decrement X co-ordinate |
| | Y2 = Y1+1 | : increment Y co-ordinate |

That is, we move either vertically to the North, or diagonally to the North East. To take any other direction would be to diverge from the path of the curve.

In order to determine which of the two courses to take, we introduce an error term, "E", and then adopt the procedure shown below:

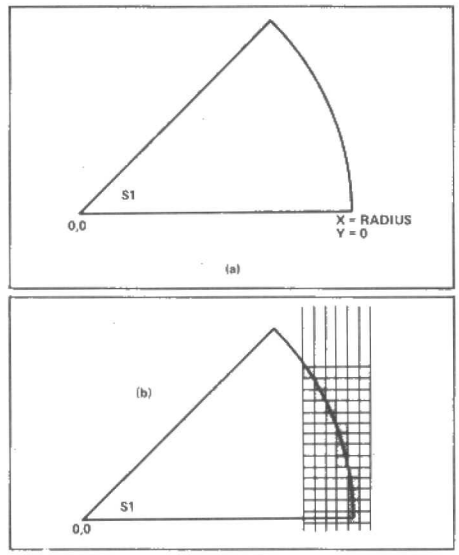



*Figure 2. The shaded area shows pixel positions superimposed on the path of the curve.*

The only real difference between them is in the way the error term, E, is dealt with. The particular version that we are going to use was chosen because it draws very "good looking" circles. It is based upon an algorithm by J. Bresenham which was originally developed for an incremental plotter and so generated a whole circle (as opposed to the one eighth slice in **Figure 2**). The error term equations for the drawing of a one eighth slice of a circle were derived by J. Michener and are shown in the BASIC program in **Listing 2** (lines 30, 110 and the seven lines shown below:

Now all 8 slices of the circle will be drawn relative to the centre co-ordinates, CX and CY.

Do not be put off by the "speed" of this BASIC version of the algorithm. The machine code implementation which follows in blindingly fast!

The full Assembly language program for the circle generator is given in **Listing 3.** As with any program of this type, a number of design decisions had to be made prior to coding which resulted in the following features:

1 A simple point clipping procedure is included in the CLPLOT subroutine (lines 1890 to 2060). Only points that lie within the screen co-ordinate range are passed to the Spectrum ROM PLOT routine. This allows us to draw circles which are only partly visible without generating any annoying error messages.

2 Two's compliment 16-bit arithmetic is used throughout. This allows us to draw the visible parts of very large circles.

3 As the ROM plotting routine corrupts one set of registers the variables CX,CY (centre co-ord's), X and Y (point co-ord's) are stored in a table within the stack. They can therefore be accessed with the POP instruction. The IX register contains the address of the base of this table.

4 The error term variable is stored in HL. The alternate register set is therefore used to update E.

5 The X co-ordinate is decremented (line 1540) before the error term is updated (as opposed to after). This results in slightly better circles than those drawn by the BASIC version of the algorithm.

6 The continual altering of the stack pointer means that the interrupts MUST be disabled.

7 The program is not relocatable. It uses absolute jumps (JP) and CALLs a built subroutine, CLPLOT. (However, see BASIC **Listing 4**).

**LISTING 2.**

```
10 INPUT "CIRCLE ";CX;" ";CY;" ";RADIUS
20 REM
30 LET E=3-2*RADIUS:   REM              Initialise error term, E.
40 REM
50 LET X=RADIUS: LET Y=0:   REM         Start at 3 o'clock position.
60 REM
65 IF Y>X THEN  STOP :   REM            Stop at end of slice.
70 REM
75 PLOT CX+X,CY+Y:   REM                PLOT point relative to
80 REM                                  centre (CX,CY) position.
90 REM
100 REM
110 IF E<0 THEN  LET E=E+4*Y+6:    REM          Check error term and update
120 IF E>=0 THEN  LET E=E+4*(Y-X)+10: LET X=X-1
130 REM                                         accordingly. Decrement X
140 REM                                         coordinate as necessary.
145 REM
150 LET Y=Y+1:   REM                    Increment Y coordinate.
160 GO TO 65
```

1 Initialise the error term, E.

2 Initialise X & Y : X = radius.
Y = 0.

3 PLOT X,Y.

4 IF E< THEN Update E
X = X (ie no change)
Y = Y+1.

5 IF E>=0 THEN Update E
X = X–1
Y = Y+1

6 IF Y<=X THEN Jump back to step 3

These six steps make up a general algorithm for drawing a one-eighth slice of a circle as in **Figure 2.** Step 6 ensures that the procedure stops at the end of the slice, where X (the horizontal distance from the centre) is equal to Y (the vertical distance from the centre).

There are several different algorithms available "off the shelf" which use the incremental method as outlined above.

The key point about these error term calculations is that they require only the simplest integer arithmetic. All the operations can be performed in Z80 machine code with little overhead.

**Listing 2** draws only one slice of a circle. To complete the picture we need to add the following seven lines:

```
76  PLOT CX+X,CY–Y
77  PLOT CX–X,CY–Y
78  PLOT CX–X,CY+Y
79  PLOT CX+Y,CY+X
80  PLOT CX+Y,CY–X
81  PLOT CX–Y,CY–X
82  PLOT CX–Y,CY+X
```

**TABLE 1.**

| Section of program | RADIUS = | 70 | 30 | 10 | Run time for 1000 circles |
|---|---|---|---|---|---|
| Whole program | | 85 | 40 | 15 | sec. |
| Without inner loop (lines 610-1300) | | 4 | 4 | 4 | sec. |
| Without Call to ROM plotting routine (line 2050) | | 25 | 17 | 7 | sec. |

These notes, together with the commentary in the listing, should make the program reasonably easy to understand.

At present, when using the routine from BASIC, the circle parameters have to be POKEd in to the appropriate locations. This is not a particularly convenient state of affairs. Those of you with an Interface 1

**LISTING 3.**

```
*HISOFT GENS3M2 ASSEMBLER*
      ZX SPECTRUM

 HISOFT 1983,4
  All rights reserved

Pass 1 errors: 00
```

```
                        10 ;****************************
                        15 ;*                          *
                        20 ;*          CIRCLE.         *
                        25 ;*    Bresenham/Michener    *
                        30 ;*        algorithm.        *
                        35 ;*                          *
                        40 ;*    Z-80 iplementation    *
                        45 ;* (c) J.R. Hibberdine 1984 *
                        50 ;*                          *
                        55 ;****************************
                        60 ;
                        70 ;
                        80 ;
                        90 ;
FA00                   100         ORG  #FA00
FA00                   110         ENT  #FA00
                       120 ;
0080                   130 CX      EQU  128
0057                   140 CY      EQU  87
0046                   150 RADIUS  EQU  70
                       160 ;
                       170 ;BEGIN
FA00 D9                180         EXX
FA01 E5                190         PUSH HL         ;SAVE HL' & IX TO ENSURE
FA02 DDE5              200         PUSH IX         ;ORDERLY RETURN TO BASIC.
                       210 ;
FA04 F3                220         DI
                       230 ;                        INTERRUPTS DISABLED TO PREVENT
                       240 ;                        STACK BEING INTERFERED WITH.
                       250 ;
FA05 018000            260         LD   BC,CX      ;CENTRE COORDINATES
FA08 115700            270         LD   DE,CY      ;OF CIRCLE.
                       280 ;
                       290 ;   SET UP PARAMETER TABLE
                       300 ;        IN STACK...
                       310 ;
FA0B 214600            320         LD   HL,RADIUS
FA0E D5                330         PUSH DE         ;CY TO STACK.
FA0F 110000            340         LD   DE,0       ;1st Y-COORD.(=0)
FA12 D5                350         PUSH DE         ;TO STACK.
FA13 C5                360         PUSH BC         ;CX TO STACK.
FA14 E5                370         PUSH HL         ;1st X-COORD.(RADIUS) TO STACK.
                       380 ;
FA15 DD210000          390         LD   IX,0
FA19 DD39              400         ADD  IX,SP      ;IX POINTS TO END POSITION
                       410 ;                       OF PARAMETER TABLE IN STACK.
                       420 ;
                       430 ;   CIRCLE ROUTINE PROPER NOW BEGINS.
                       440 ;  FIND FIRST 'E' VALUE..
FA1B E5                450         PUSH HL         ;RADIUS
FA1C D9                460         EXX
FA1D E1                470         POP  HL
FA1E 29                480         ADD  HL,HL      ;2*RADIUS
FA1F EB                490         EX   DE,HL
FA20 210300            500         LD   HL,3
FA23 A7                510         AND  A
FA24 ED52              520         SBC  HL,DE      ;E=3-2*RADIUS.
FA26 D9                530         EXX
                       540 ;
                       550 ;   NOW ENTER INTO MAIN LOOP...
                       560 ;
FA27 A7                570 LOOP    AND  A
FA28 ED52              580         SBC  HL,DE      ;IF Y>X THEN
FA2A FA9DFA            590         JP   M,EXIT     ;    EXIT FROM LOOP.
                       600 ;
FA2D 3E02              610         LD   A,2        ;INNER LOOP COUNTER.
FA2F 08                620 LPI     EX   AF',AF
                       630 ;   THE CODE WITHIN THIS LOOP CALCULATES AND PLOTS THE
                       640 ;  SCREEN COORD.S FOR EACH CIRCLE SEGMENT.
                       650 ;
                       660 ;   THE COORD.S ARE DEALT WITH IN TWO LOTS OF 4 :-
                       670 ;         1st TIME THRU LOOP              2nd TIME...
                       680 ;             CX+X,CY+Y                   CX+Y,CY+X
                       690 ;             CX+X,C-Y                    CX+Y,CY-X
                       700 ;             CX-X,CY+Y                   CX-Y,CY+X
                       710 ;             CX-X,CY-Y                   CX-Y,CY-X
                       720 ;
                       730 ;  NOTE THAT AT THIS STAGE SP POINTS TO
                       740 ; THE END OF THE PARAMETER TABLE.
                       750 ;
FA30 D1                760         POP  DE         ;X(1st TIME) Y(2nd TIME)
FA31 E1                770         POP  HL         ;CX
FA32 19                780         ADD  HL,DE
FA33 EB                790         EX   DE,HL
FA34 C1                800         POP  BC         ;Y(1st) X(2nd)
FA35 E1                810         POP  HL         ;CY
FA36 09                820         ADD  HL,BC
FA37 DDF9              830         LD   SP,IX      ;STACK RESET. SP NOW POINTS
                       840 ;                        TO END OF PARAMETER TABLE.
FA39 CDA8FA            850         CALL CLPLOT
                       860 ;
FA3C D1                870         POP  DE         ;X(1st) Y(2nd)
FA3D E1                880         POP  HL         ;CX
FA3E 19                890         ADD  HL,DE
FA3F EB                900         EX   DE,HL
FA40 C1                910         POP  BC         ;Y(1st) X(2nd)
FA41 E1                920         POP  HL         ;CY
FA42 A7                930         AND  A
FA43 ED42              940         SBC  HL,BC
FA45 DDF9              950         LD   SP,IX      ;STACK RESET.
FA47 CDA8FA            960         CALL CLPLOT
                       970 ;
FA4A D1                980         POP  DE         ;X(1st) Y(2nd)
FA4B E1                990         POP  HL         ;CX
FA4C A7               1000         AND  A
FA4D ED52             1010         SBC  HL,DE
FA4F EB               1020         EX   DE,HL
FA50 C1               1030         POP  BC         ;Y(1st) X(2nd)
FA51 E1               1040         POP  HL         ;CY
FA52 09               1050         ADD  HL,BC
FA53 DDF9             1060         LD   SP,IX      ;STACK RESET.
FA55 CDA8FA           1070         CALL CLPLOT
                      1080
FA58 D1               1090         POP  DE         ;X(1st) Y(2nd)
FA59 E1               1100         POP  HL         ;CX
FA5A A7               1110         AND  A
FA5B ED52             1120         SBC  HL,DE
FA5D EB               1130         EX   DE,HL
FA5E C1               1140         POP  BC         ;Y(1st) X(2nd)
FA5F E1               1150         POP  HL         ;CY
FA60 A7               1160         AND  A
FA61 ED42             1170         SBC  HL,BC
FA63 DDF9             1180         LD   SP,IX      ;STACK RESET.
FA65 CDA8FA           1190         CALL CLPLOT
                      1200 ;
FA68 C1               1210         POP  BC
FA69 E1               1220         POP  HL
FA6A D1               1230         POP  DE
FA6B C5               1240         PUSH BC
FA6C E5               1250         PUSH HL         ;X&Y ARE
FA6D D5               1260         PUSH DE         ;SWAPPED IN TABLE.
                      1270 ;
FA6E 08               1280         EX   AF',AF
FA6F 3D               1290         DEC  A
FA70 C22FFA           1300         JP   NZ,LPI     ;GO BACK TO PLOT POINT
                      1310 ;                        ;IN OTHER 4 SEGMENTS.
FA73 C5               1320         PUSH BC         ;SAVE Y.
                      1330 ;
                      1340 ;   HAVING PLOTTED ONE POINT
                      1350 ;  IN EACH OF THE 8 SEGMENTS
                      1360 ;  WE NOW TEST & UPDATE THE ERROR TERM 'E'...
                      1370 ;
FA74 D9               1380         EXX
FA75 EB               1390         EX   DE,HL
FA76 CB7A             1400         BIT  7,D        ;TEST SIGN OF ERROR TERM 'E'.
FA78 CA84FA           1410         JP   Z,POS
                      1420 ;
                      1430 ;   IF E<0 THEN...
FA7B E1               1440         POP  HL         ;Y
FA7C 29               1450         ADD  HL,HL      ;*2
FA7D 29               1460         ADD  HL,HL      ;Y*4
FA7E 010600           1470         LD   BC,6
                      1480 ;                ...E=E+4*Y+6
FA81 C390FA           1490         JP   J1         ;JUMP AHEAD TO DO FINAL ADDITIONS.
                      1500 ;
                      1510 ;   IF E>=0 THEN...
FA84 E1               1520 POS     POP  HL         ;Y
FA85 C1               1530         POP  BC         ;X
FA86 0B               1540         DEC  BC         ;X=X-1
FA87 C5               1550         PUSH BC         ;NEW X-COORD VALUE TO TABLE
                      1560 ;
FA88 A7               1570         AND  A
FA89 ED42             1580         SBC  HL,BC
FA8B 29               1590         ADD  HL,HL
FA8C 29               1600         ADD  HL,HL      ;(Y-X)*4
FA8D 010A00           1610         LD   BC,10
FA90 09               1620 J1      ADD  HL,BC
FA91 19               1630         ADD  HL,DE
                      1640 ;              ...E=E+(Y-X)*4+10
                      1650 ;
                      1660 ;   NOW INCREMENT Y-COORD VALUE...
                      1670 ;
FA92 D9               1680         EXX
FA93 E1               1690         POP  HL         ;X
FA94 D1               1700         POP  DE
FA95 D1               1710         POP  DE         ;Y
FA96 13               1720         INC  DE         ;Y=Y+1
FA97 D5               1730         PUSH DE
FA98 DDF9             1740         LD   SP,IX      ;STACK RESET & TABLE UPDATED.
                      1750 ;
FA9A C327FA           1760         JP   LOOP       ;DO ALL OVER AGAIN!!
                      1770 ;
FA9D 210800           1780 EXIT    LD   HL,8       ;CLEAR
FAA0 39               1790         ADD  HL,SP      ;PARAMETER TABLE
FAA1 F9               1800         LD   SP,HL      ;FROM STACK.
                      1810 ;
FAA2 DDE1             1820         POP  IX         ;IX & HL' RESTORED, AND INTERRUPTS
FAA4 E1               1830         POP  HL         ;ENABLED FOR RETURN TO BASIC
FAA5 D9               1840         EXX
FAA6 FB               1850         EI
FAA7 C9               1860         RET
                      1870 ;
                      1880 ;
                      1890 ;   PLOT SUBROUTINE..
                      1900 ;  WITH SIMPLE POINT CLIPPING PROCEDURE.
                      1910 ;
22E5                  1920 PLOT    EQU  #22E5      ;16K ROM PLOT ROUTINE.
                      1930 ;
FAA8 7A               1940 CLPLOT LD   A,D
FAA9 A7               1950         AND  A
FAAA C0               1960         RET  NZ         ;X>25 OR X<0.5
```

**LISTING 3 – Continued**

```
FAAB 7C      1970      LD   A,H
FAAC A7      1980      AND  A
FAAD C0      1990      RET  NZ        ;Y>255 OR Y<0.
FAAE 3EAF    2000      LD   A,175
FAB0 BD      2010      CP   L
FAB1 D8      2020      RET  C         ;Y>175
FAB2 45      2030      LD   B,L       ;Y-COORD.
FAB3 4B      2040      LD   C,E       ;X-COORD.
FAB4 CD6522  2050      CALL PLOT
FAB7 C9      2060      RET
```

```
Pass 2 errors: 00

CLPLOT FAA8    CX     0080
CY     0057    EXIT   FA9D
J1     FA90    LOOP   FA27
LPI    FA2F    PLOT   22E5
POS    FA84    RADIUS 0046

Table used:   119  from   200
Executes: 64000
```

could set up a new CIRCLE command as described in Ian Logan's *Microdrive Book*. As an alternative (and this applies to those of you without an Interface 1), you could use a rather elegant approach to parameter passing which was described in an article by Mike James in the May '84 edition of *Electronics and Computing Monthly*. Basically, this method makes use of the way in which the Spectrum passes parameters to user defined function. By setting up such a function as follows:

```
DEF FN C(X,Y,R) = USER 64000:
REM Address of routine.
```

and then replacing lines 260, 270 and 320 of **Listing 3** with the code:

```
275   LD   IX,(#5C0B)   ;This address
280;                     contains
285;                     system
290;                     variable
                         DEFADD. It
                         points to
                         current user
                         defined
                         function.
295   LD   C,(IX+4)     ;CX
300   LD   B,(IX+5)
305   LD   E,(IX+12)    ;CY
310   LD   D,(IX+13)
315   LD   L,(IX+20)    ;RADIUS
320   LD   H,(IX+21)
```

We can now access the circle routine from BASIC with the statement

```
RANDOMISE FN C(CX,CY,RADIUS)
```

The parameters will then be passed to the routine and the circle drawn.

When using this method you should remember two things:

1 The variable names within the DEF FN statement MUST be single letters.

2 The variables within the function call (ie FN C(CX,CY,RADIUS)) MUST be integers within the range –32768 to +32768.

For more details of this method you should refer to the original article.

The BASIC program in **Listing 4** contains all the information required for non-speakers of Z80 Assembly language to implement the circle generator in machine code. Of course, some careful typing is required but the effort is well worth it. The program will also relocate the code to wherever you want; all that's needed is 197 empty bytes. As it stands, the code is configured to execute from its START address, 64000 (#FA00), so if that is a satisfactory arrangement, you do not have to enter lines 5 and 70 to 150. The program also implements the parameter passing technique as described earlier. Lines 160 to 170 give a brief demonstration of how everything works from BASIC.

As you would expect, this machine code routine is much faster than the BASIC CIRCLE command. It will draw 1000 circles (radius=70) in approximately 85 seconds. In contrast, a CIRCLE 128,87,70 command, executed 1000 times takes 15 minutes!

The program is the usual compromise between space, speed and clarity, and so it could probably be persuaded to run a little faster. However, those of you who want to make the routine really fly should take note of the timing figures in **Table 1:**

On average, between one half and two thirds of the execution time is taken up by the PLOT routine in ROM. Therefore, one way to increase the speed would be to design a faster point plotting procedure. As an example, you might consider the severely curtailed version of PLOT below which replaces the CALL PLOT instruction (line 2050) in the program.

```
      CALL  #22AA  ;PIXEL ADDRESS
                    routine in ROM.
                   ;Returns address in
                    HL & A.
      LD    B,A
      INC   B      ;Position of pixel
                    within byte (HL).
      LD    A,1    ;Plotted point.
LP    RRCA         ;Position point
      DJNZ  LP     ;correctly within the
                    byte
      LD    B,(HL) ;Get relevant byte
                    from screen.
      OR    B      ;Superimpose point
                    from A.
      LD    (HL),A ;Put byte back on to
                    screen.
```

With this code in place the "1000 circles" loop (radius=70) takes 55 seconds; a considerable improvement of course, you have to pay for what you get; no INVERSE OVER or local PAPER and INK commands available.

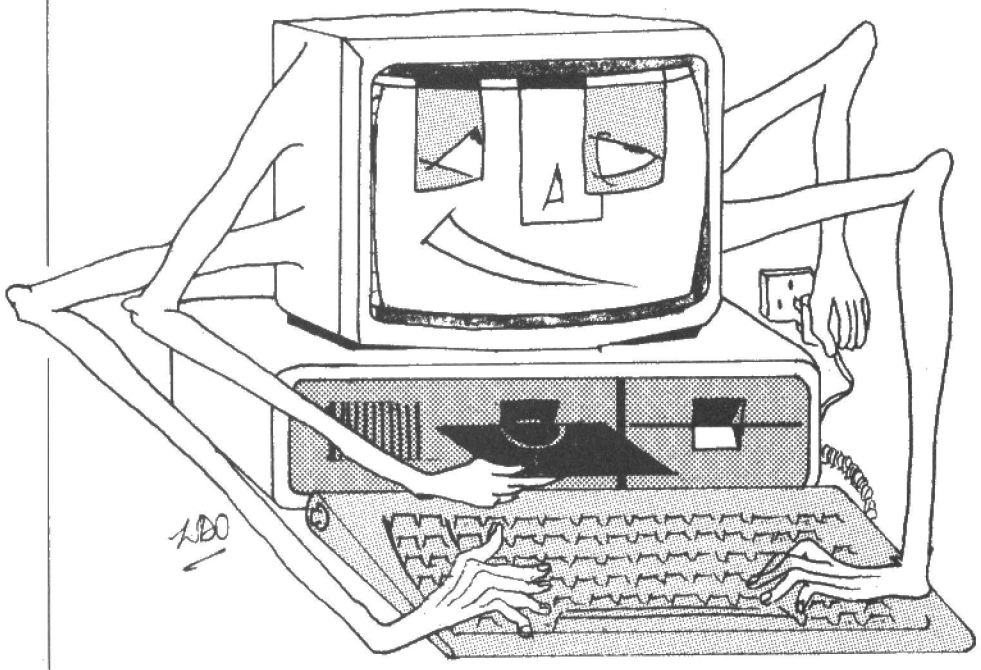
So that's it! You can now draw circles much more quickly than you could before. Indeed, you might even say that this routine runs rings around. . .

**LISTING 4.**

```
  5 INPUT "START ADDRESS ";START
 10 RESTORE 295
 20 FOR C=0 TO 196
 30 READ B
 40 POKE START+C,B
 50 NEXT C
 60 REM RELOCATE..
 70 RESTORE 140
 80 FOR I=1 TO 9
 90 READ A,K
100 LET A=A+START: LET K=K+START
110 POKE A,K-256*INT (K/256)
120 POKE A+1,INT (K/256)
130 NEXT I
140 DATA 56,170,71,181,85,181,99,181,115,181
150 DATA 126,60,134,145,143,157,168,52
155 REM ***************
156 REM ******DEMO******
160 DEF FN C(X,Y,R)=USR START
162 FOR C=0 TO 150 STEP 10
164 RANDOMIZE FN C(128,87,C)
166 NEXT C
168 PRINT #1;"Pretty good, eh?": PAUSE 0
169 REM ***************
170 STOP
180 REM ***************
240 REM ***************
260 REM DECIMAL OPCODES...
295 DATA 217,229,221,229,243,221
297 DATA 42,11,92,221,78,4
298 DATA 221,70,5,221,94,12
300 DATA 221,86,13,221,110,20
301 DATA 221,102,21,213,17,0,0
303 DATA 213,197,229,221,33,0
304 DATA 0,221,57,229,217,225
305 DATA 41,235,33,3,0,167
306 DATA 237,82,217,167,237,82
307 DATA 250,157,250,62,2,8
308 DATA 209,225,25,235,193,225
309 DATA 9,221,249,205,168,250
310 DATA 209,225,25,235,193,225
311 DATA 167,237,66,221,249,205
312 DATA 168,250,209,225,167,237
313 DATA 82,235,193,225,9,221
314 DATA 249,205,168,250,209,225
315 DATA 167,237,82,235,193,225
316 DATA 167,237,66,221,249,205
317 DATA 168,250,193,225,209,197
318 DATA 229,213,8,61,194,47
319 DATA 250,197,217,235,203,122
320 DATA 202,132,250,225,41,41
321 DATA 1,6,0,195,144,250
322 DATA 225,193,11,197,167,237
323 DATA 66,41,41,1,10,0
324 DATA 9,25,217,225,209,209
325 DATA 19,213,221,249,195,39
326 DATA 250,33,8,0,57,249
327 DATA 221,225,205,217,251,201
328 DATA 122,167,192,124,167,192
329 DATA 62,175,189,216,69,75
330 DATA 205,229,34,201
```

# MY KINGDOM FOR AN OS

## William Owen sheds light on the (user) transparent software that can mean KOS or QDOS for the ever suffering programmer.



At least five versions of QDOS, the vaingloriously named operating system for the Sinclair QL, are known to exist. Anyone who has heard of the immense problems surrounding the development of QDOS will see the joke when they discover that there is yet another. Unbeknown to the Sinclair marketing department, QDOS wasn't the first systems software to receive the acronym. The Quick and Dirty Operating System (no relation) was a prototype of Microsoft's MS DOS.

The advent of the cheap floppy disc drive has been accompanied by a new generation of apparently arbitrary sets of initials with which to wrestle. To control the means of mass data storage now available, QL, BBC micro, Spectrum and Commodore owners must face up to the challenges of the operating system, of QDOS, CP/M, SP DOS and the DFS.

Initially very limited system control commands were available to or needed by the home computer user. The Basic interpreter commands such as LOAD, SAVE, OPEN or CLOSE provoke a call to a routine in ROM which executes a simple I/O function. Other routines deal with control of the keyboard and screen handling, usually in a manner invisible to the user.

A primitive control system of this kind could, with licence, be described as a basic operating system. It is adequate for controlling cassette recorder storage but not for controlling disc drives, allows only limited I/O, printer or program control, and is not much help to the programmer or user.

At the other end of the scale business users can buy uncomplicated systems which cope with many operators and many programs simultaneously; offer facilities for showing more than one application running on the screen at one time; and make program access and use childs play, as well as performing all the housekeeping duties required of the systems software. But the new technology is not restricted to those with bottomless wallets. It has filtered down to the home user in the form of multitasking OS9 on the Dragon 64, QDOS windows on the Sinclair QL, and the slower but surer BBC DFS. New systems pose new questions. What is an operating system? How does it work? And how can I get the best out of it?

The most important function of an operating system (OS) is as an interface between the user and the computer. In particular, handling the keyboard, the screen (or console), the printer, disc drives, and to act as an interpreter of commands.

The operating system acts as a link and a controller between the components of the computer, and in particular between the heart of the machine – the CPU – and the various appendages through which its power becomes manifest. Because the computer is a literal minded machine it must be led by hand, step-by-step, through each process that it is required to perform. Obviously no user wants to have to tell the CPU how to format or access a disc each time the task is to be performed. The details of how the operation is carried out are handled by the OS; the user merely tells the computer, via the OS, which task is next on the agenda.

### Primary functions

An operating system is therefore a control program, providing the user with a flexible and manageable means of control over the resources of the computer. Within that control program are sets of instructions for fulfilling all of the common processes used by the computer to complete a task. The primary processes or functions can be divided into three groups:

- To provide an orderly and consistent input/output capability for the various elements of the computer (console, keyboard, printer, hard or floppy disc, tape, bus interface etc.).
- To provide file management and status reporting for the data stored in the computer system. A file management system allows the user to find out what files are on a disc, how big they are, how much unused space is left, as well as managing the reading and writing of information to and from a disc.
- To provide for the loading and execution of user programs.

**"... Sinclair's QDOS has an early Microsoft namesake ..."**

To make that sound a little more impressive, let's consider the practical implications for the user of applications software and for the programmer.

The programmer is given two major advantages. Operating systems interpret machine code commands and in basic form are written for processors, not for individual machines. CP/M for example is in its most common guise an 8-bit OS found on Z80 and 8080 based machines. It is therefore an easy task to transport a program from one machine to another, as long as they use the same OS. A certain amount of reformatting is required to take into account any particular or peculiar hardware and I/O characteristics, but that is a small job compared to rewriting every call to I/O, disc drive or console, and

changing the allocation of memory.

The second advantage relates to the way in which the OS acts as a link between the machine and the applications software. Without an OS the programmer must consider the way in which each aspect of the computer and its peripherals interact with the software he or she is writing. For example, if a word processor needs to send a message to the printer, an appropriate command is issued and converted by the language processor to a set of machine code instructions, including a call to the OS. It is the OS which decides which port is assigned to the printer, checks its status, and prepares the relevant message to the hardware.

As far as the user of applications software is concerned, the less seen of the OS the better – the OS is there to make life easier and why look under the bonnet when you don't have to? At one extreme a program could be written to by-pass the OS prompt, access a file on the disc automatically, and take the user straight into a wordprocessor or database. But to show the benefits of using an OS, the following is a small selection of the memory management facilities commonly available:

An operating system will allocate a space on the disc for a data file and decide how much space on the disc is required; keep a directory of all files on a disc; copy a file or set of files from one disc to another; compare two files; automatically date stamp a file; sort files into a predetermined sequence; recover damaged files; tell the user how long a file is and how much storage space is left on the disc. To initiate most of these tasks all that is required of the user is a simple one word or one line statement.

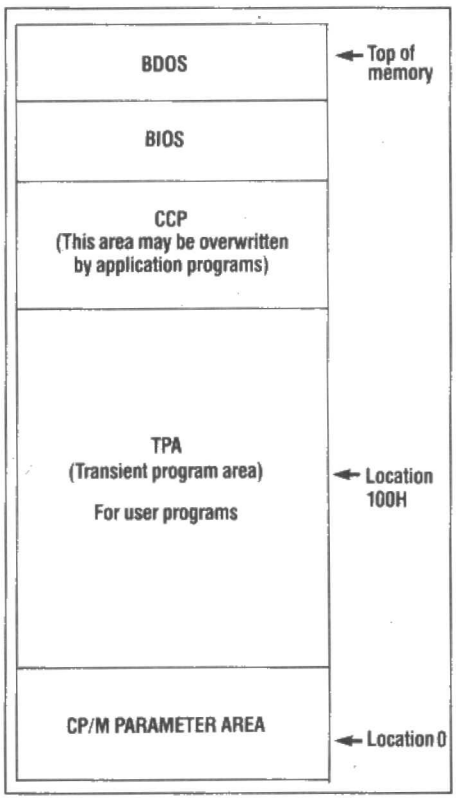



*Figure 1. CP/M memory map.*

## CP/M – the old faithful

For a more detailed examination of the functions and capabilities of an operating system, it is worth taking a look at one popular system. CP/M is a good example because it exists in both 8-bit and 16-bit versions, and is probably the most widely used OS in the world today.

The history of CP/M (it stands for Control Program for Microcomputers) parallels that of the microcomputer. The first 4-bit microprocessor was available from 1970, but it was not until 1973 that a device powerful enough to be used within a microcomputer arrived; that was the Intel 8080. Early micro users were in a very similar situation to the majority of home computer users today. There were no operating systems, no disc drives. Cassette recorders were used for data and program storage.

A programmer working for Intel named Gary Kildall was one of the first people to get hold of the floppy disc drives which appeared in 1974, and soon realised that some form of file manager was necessary to control the writing and reading of data. Kildall set about writing a disc controller program – the prototype of CP/M's BDOS – and, when Intel adopted a different system, left to set up Digital Research to market his own product.

The next step was to consolidate all of the hardware dependent portions of CP/M in one section (the BIOS) so that anyone could buy a copy of the OS and do their own modifications. It was therefore possible to adapt the system to any micro, and to develop sophisticated software in the CP/M environment, in the sure knowledge that it would run on the hardware products of a wide range of manufacturers. At various stages an assembler, debugger editor, and number of system utilities were added to enable the user to write programs, store and retrieve data, and in general utilise the full capability of the microcomputer. Millions of copies of CP/M were sold and it rapidly became the universal standard for 8-bit micros.

In all CP/M computers the maximum amount of memory which the CPU can address is 64K. The OS divides the available memory into a number of distinct blocks and reserves specific areas for its own use. CP/M locations are reserved in the lowest portion of memory, so that a CP/M system can be run on a micro containing between 32K and 64K main memory.

**"... new developments include multi-user, multi-tasking and windowing ..."**

CP/M is split up into three major subsections: the Basic Disc Operating System (BDOS); the Basic Input/Output System (BIOS) and the Console Command Processor (CCP). These sections sit at the top of memory, above the Transient Program Area (TPS); which is reserved for user programs. The TPA sits above the area reserved for CP/M (see **Figure 1**).

BDOS is the portion of CP/M which handles all of the basic disc file transactions, such as reading and writing a record to and from a disc, the dynamic allocation and de-allocation of disc space, and other disc oriented tasks. BDOS, unlike BIOS, is entirely machine independent; in other words, BDOS is the same for all micros regardless of the particular disc interface or the combination of I/O devices hooked up to the computer. BDOS can be considered to be the core of CP/M.

**"... an OS is the link between the heart of a computer, CPU, and the peripherals that endow it with power ..."**

BIOS is the second major subsection of the OS. BIOS contains all of the programming in CP/M that is machine dependent. Thus it is in BIOS that all I/O programs (such as those for the console device, the disc controller interface, and the printer) are contained. It is through the BIOS/BDOS concept that CP/M achieves its wide range of potential host machines. BDOS and BIOS make up the nucleus of CP/M, and are always present whether an applications program of CP/M utility is being run.

CCP is the portion of CP/M which controls interaction with the user in command mode, ie when a user program is not running and is in control of the computer. It allows programs to be loaded into the TPA and run, files to be listed, created, deleted, and other housekeeping functions. CCP is the one area of CP/M which can be overwritten by applications programs.

The TPA is the area in main memory, between location 256 (the top of the lower memory area reserved for CP/M) and the bottom of BDOS. User programs are stored and executed under CP/M within the TPA. CP/M loads user programs into this area starting at location 256 and working upwards, filling bytes until either the program is completely loaded or the system is out of available memory. The TPA's size varies with the amount of available RAM in the computer, whereas BDOS, BIOS and CCP are fixed in size. If necessary, CP/M will overlay an applications program over the CCP in order to provide sufficient memory space. When the program has finished and control is to be returned to the CCP, the CCP is reloaded from the disc as a 'warm boot'. A cold boot reads in the whole CP/M system software, but during a warm boot the rest of CP/M is assumed to be intact as there has been no power down.

An applications program communicates with the various hardware devices of a

computer via CP/M, by a series of standard protocols or commands known as system calls. System calls are routines which perform specific 'low-level' functions. An 8080 CALL instruction, sent from the program, initiates the desired routine. System calls are used in the program whenever control of the computer is passed over to CP/M to accomplish a specific task. Control is returned to the program by the OS when the task is complete. For example, assuming a program wants to output a character to the current console device, at the appropriate output point in the program the character and command are passed to CP/M along with control of the computer. When CP/M returns from the system call to the program the character will have been output to the console device (eg the screen).

Most CP/M operations are of course invisible to the user. What the user sees is the 'shell' of the program, ie the prompts which appear on the screen indicating, for example, which disc drive is in use, or system errors. CP/M 80 (the 8-bit version) supports up to 15 disc drives (labelled A to P). To change the logged drive the user simply answers the prompt with the allocated letter followed by a colon.

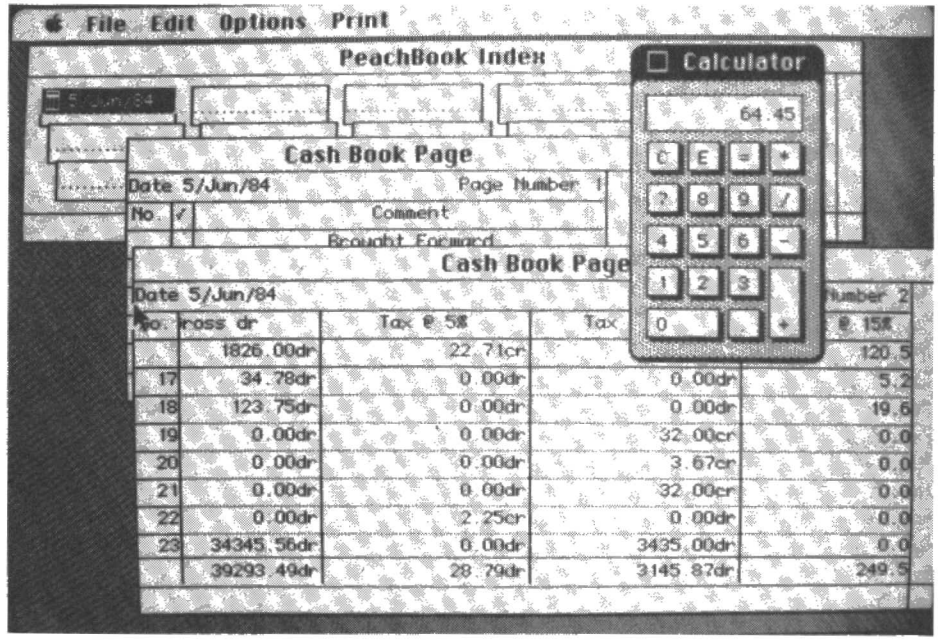
Not every operation in CP/M is that easy. Certain aspects of the system could by no means be described as user-friendly: how many people know what to do when confronted with B:DATE.CMD set to directory(DIR), road only(RO)? CP/M may have over 50% of the operating systems market but it is now most definitely obsolete.

## New concepts

Two years ago the dominant position of CP/M was hijacked by Microsoft when IBM adopted PC DOS, a version of Microsoft MS DOS, for the best selling IBM PC. Some say that MS DOS is a slavish imitation of CP/M; the command structure bears a similar dissimilarity to standard English. MS DOS and PC DOS offer improvements to the way disc files are structured, using a tree or root directory in which files are divided into sections and subsections for easy retrieval and general handling.

MS DOS and its 16-bit CP/M equivalent, CP/M 86, are already due for the shelf. In their place come three new developments, usually found on the 32/16 bit 68000 systems, which make computer control either more comprehensive or more comprehensible: they are multi-user or networking systems; multi-tasking systems, otherwise known as concurrent; and the closely related concept of windowing.

UNIX is a multi-user, multi-tasking system tipped as the CP/M of tomorrow. UNIX was developed for use on mainframe computers at the Bell Telephone research laboratories in the United States. Multi-user systems enable more than one user to use a processor at one time. In other words the system will have two or more terminals, one processor, and a minimum of one or more printers and hard or floppy disc storage systems. Processor, disc and printing time is allocated to the user by the OS. The success of UNIX has been assured now that it has been adopted by IBM for the PC AT – the networking, multi-tasking hard disc version of the PC.




*Screen shot of MacAccounting running on Apple's Macintosh computer and illustrating the way in which windows can be used to display many different items of information simultaneously.*

## QL multitasking

We can come back down to the affordable earth by taking a look at multi-tasking on the Sinclair QL. QDOS multi-tasking allows the user to run several (up to 60) independent programs together. Each task is known as a job: a program unit which can operate in isolation from other program units once it is assigned a particular job number. Jobs can be given different priorities, to operate in the background (printing, for example) or foreground (probably an application such as wordprocessing). A low priority job can be called up by the user whenever it is required.

## "UNIX, the CP/M of tomorrow"

Getting 10 jobs to run together is not necessarily a difficult business. On a micro such as the BBC with a simple disc filing and I/O system the user would have to intercept numerous interrupts. On the QL, this is all taken care of by QDOS as long as the correct system calls are included within the programs. To ensure that all jobs can happily run together the OS allocates different areas of memory to different tasks and correctly allocates machine resources. Each job may appear to run simultaneously, but in fact the processor is constantly switching from one to another, or will call up one job under instruction from the user while maintaining another in operation in the background. This process is invisible to the user, and to the programmer, and ensures that the execution of one job does not interfere with that of another until it is finished.

## The transparent OS

QDOS also has a limited version of a facility known as windowing. On the QL the user can watch the execution of more than one program on the screen at one time. Windowing in the correct sense of the term does much more, and is an innovation of immense importance because it shows the true potential of the operating system as an interpreter between the computer and the user.

Microsoft, Apple and Digital Research have each developed a version of windowing; all three work on similar principles and are 99% idiot proof. Briefly, windowing is a tiling or overlapping system whereby on request the OS overlays the screen with a different application. The system has a set of standard data exchange formats to enable applications to communicate, even if developed independently.

The real beauty of windowing is that the user does not have to learn any system commands. A multiple choice of commands (not as acronyms or abbreviations, but in the English we know and love) are presented to the user, and one is picked out and trapped by using the cursor (usually controlled by a mouse). Many options are presented as icons, for example the system is booted up by trapping a representation of disc, or matter is deleted by choosing a wastepaper bin. Screens are scrolled, text moved, menus pulled down from the top of the screen, by physically moving the mouse around.

Those who have agonised over the intricacies of CP/M may feel cheated by the ease with which it is now possible to learn how to control a computer. But the hackers should not be too upset. They too will be able to delve their way faster and further into the CPU, and unless they are feeling exceptionally DIM or PEEKy can forget about those endless vectors, interrupts and data statements.

# BLOWING YOUR OWN PART TWO

## Andy Green concludes the description of his fast BBC EPROM blower with a full listing of the project's comprehensive control software.

Last month we described the building and testing of the EPROM Blower's electronics – that's only half the story though. This month we bring things to their natural conclusion by showing how to do what it was designed for rather than simply looking pretty.

Having verified operation with **Listing 2** (shown last month) it is time for the acid test. The best way to accomplish this is to take a programmed ROM/EPROM and, using the sideways ROM sockets of the Beeb, compare it with what the programmer says.

Assuming a suitable ROM is installed in the micro, call the test program by typing "G.30000". You will then be asked for a ROM socket number. The vacant sockets within the BBC micro are numbered from 15, rightmost when looking from the front of the machine, to 12. If any of the commercially available sidways ROM expansion systems are fitted, refer to the board's handbook in order to determine the correct number to enter for a special socket. Type in the appropriate number and hit return. After a short pause, the prompt will reappear and a copy of the ROM will now be in memory from &3000 onwards. If it is an 8K ROM "SAVE COPY 3000 5000", and if it is a 16K ROM ""SAVE COPY 3000 7000".

Now switch off the BBC micro and take out the newly copied ROM. Load the EPROM control program and execute it. Now plug the EPROM removed from the Beeb into the ZIF socket of the programmer and use option 2 to load the file "COPY" into memory.

Now option 4 may be used to compare what is in the memory buffer with the contents of the EPROM. If this test fails then use option 4 to load into memory what the programmer thinks is there and look at it using the inbuilt examiner. Pres <SPACE> to get back to the menu and the cursor keys to move up/down.

## In control

The control program, **Listing 3,** has a comprehensive number of options. These are as follows:

1) Toggle between 2764 (8K) and 27128 (16K). A self-explanatory option, – the current selection is displayed on-screen.

2) Load in a file. This loads a file from disk/cassette into the buffer memory.

3) Check for erased EPROM. Checks for &FF in all EPROM locations.

4) Load in from EPROM. Reads EPROM

**LISTING 3. The control program.**

```
   10 REM EPROM CONTROLLER V2.1
   50 DIM F% 500:E%=&20:B%=&3000
   60 ads=&70:bits=&72:cou=&73:buf=&74:1
temp=&76:pwr=&77
   90 PROCasm:MODE7
  100 ?&FE6C=&CE:VDU23;8202;0;0;0;:PROCp
age("MAIN MENU")
  110 PRINT"  1) TOGGLE BETWEEN 2764/271
28"''"  2) LOAD IN A DISC FILE TO BUFFER
"''"  3) CHECK FOR ERASED EPROM"''"  4)
LOAD IN EPROM CONTENTS"''"  5) BLOW BUFF
ER INTO EPROM"''"  6) EXAMINE BUFFER CON
TENTS"''"  7) COMPARE BUFFER WITH EPROM"
  115 PROCgetcom(7)
  120 ON C%GOTO200,210,250,300,350,400,5
50
  200 IFE%=&40THEN E%=&20 ELSE E%=&40
  205 PROCupd:GOTO115
  210 PROCpage("LOADING A FILE TO BUFFER
"):INPUT "Type the filename: " A$:H%=&C0
0:$H%="LOAD "+A$+" 3000"+CHR$13:X%=0:Y%=
12:CALL&FFF7:GOTO100
  250 PROCpage("EPROM ERASURE TEST"):A%=
2:L%=E%:E%=&40:CALLrd:E%=L%:IF(?bits)=0T
HENPRINTTAB(12,13);"²EPROM IS ERASED" EL
SE PRINTTAB(13,13)"¹NO GO AT &";~(256*?a
ds)+(ads?1)
  260 GOTO360
  300 A%=0:CALLrd:GOTO400
  350 PROCpage("BURNING THE EPROM"):CALL
burn:IF(?bits)=0THENPRINTTAB(6,13);"²ALL
 DATA VERIFIED OK!!!!!" ELSE PRINTTAB(0,
13);"¹ BAD DATA @ &";~((!buf)AND&FFFF);"
 prog'd &";~?&80;", read &";~?&81
  360 PRINTTAB(0,24);CHR$131+CHR$157+CHR
$132+"  Press any key to return to menu"
;:A%=GET:GOTO100
  400 CLS:H%=&3000:PRINTTAB(0,12);:FORT%
=H%TOH%+96STEP8:PROC1(T%):NEXT:*FX4,1
  405 *FX12,12
  410 Z%=0:REPEAT:A%=GET:IFA%<>139THEN45
0
  420   K%=H%-96:IFK%<&2FA0THEN500 ELSE
VDU30,11:H%=H%-8:IFK%<&3000THEN500
  430   PROC1(K%):GOTO500
  450   IFA%<>138THEN490
  460   K%=H%+104:IFK%>&7060THEN500 ELSE
 PRINTTAB(0,24);:H%=H%+8:IFK%>&7000THEN5
00
  470   PROC1(K%):GOTO500
  490   IFA%=32THENZ%=1
  500   UNTILZ%:GOTO100
  550 PROCpage("CHECKING EPROM AGAINST B
UFFER"):A%=1:CALLrd:IF(?bits)=0THENPRINT
TAB(11,13);"²CHECKS OUT EXACTLY" ELSE PR
INTTAB(13,13)"¹NO GO AT &";~(256*?ads)+(
ads?1);TAB(10,15);"Buff= &";~?&81;" EPRO
M &";~?&80
  560 GOTO360
  999 STOP
 1000 DEFPROCpage(A$):CLS:A$=CHR$132+CHR
$157+CHR$135+CHR$141+STRING$((34-LENA$)D
IV2," ")+A$:PRINTA$'A$:PROCupd:ENDPROC
 1010 DEFPROCupd:PRINTTAB(0,3);CHR$129+C
HR$157+CHR$135;"&3000";TAB(28,3);:IFE%=&
40THENPRINT;"27128 (16K)"; ELSEPRINT;"27
64  ( 8K)";
 1020 PRINT''':ENDPROC
 1030 DEFPROCgetcom(A%):PRINTTAB(0,24);C
HR$131+CHR$157+CHR$132+"    Press the ap
propriate number";:REPEAT:VDU7:C%=GET-48
:UNTIL(C%>0)AND(C%<=A%):ENDPROC
 1040 DEFPROC1(A%):PRINT;~A%;"  ";:FORY%
=0TO7:K%=Y%?A%:IFK%<16THENVDU48
 1050   PRINT;~K%;" ";:NEXT:PRINT"  ";:F
ORY%=0TO7:K%=(Y%?A%)AND127:IF(K%>31)AND(
```

**LISTING 3 – Continued**

```
K%<>127)THENVDUK% ELSE VDU46
 1060   NEXT:ENDPROC
 2000 DEFPROCasm:FORY%=0TO3STEP2:P%=F%:[
OPTY%:.init LDA#0:STApwr:LDA#&50:JSRlatc
h:LDA#&51:JSRlatch:LDA#&50:JMPlatch
 2020   .latch STAltemp:LDApwr:BEQlaq:LD
A#&88:ORAltemp:STAltemp:.laq LDA#&FF:STA
&FE62:LDAads:AND#16:LSRA:LSRA:LSRA:LSRA:
ORAltemp:STAltemp:LDAads:AND#32:ORAltemp
:STA&FE60:LDA#&DE:STA&FE6C:LDA#&FE:STA&F
E6C:RTS
 2050   .rd STAbits:LDA#0:STAbuf:STAads:
LDA#&30:STAbuf+1:LDA&414:STAcou:LDA#&50:
JSRlatch:LDA#&54:JSRlatch:LDA#&50:JSRlat
ch:LDY#0
 2060   .r1 JSRadl:LDA#&10:JSRlatch:LDA#
0:STA&FE62:LDAbits:BNEcom:LDA&FE60:STA(b
uf),Y:JMPend:.com CMP#2:BEQera:LDA&FE60:
CMP(buf),Y:BEQend:.ngo STYads+1:STA&80:L
DA(buf),Y:STA&81:LDA#&FF:STAbits:RTS:.er
a LDA&FE60:CMP#&FF:BNEngo
 2070   .end LDA#&52:JSRlatch:INY:BNEr1:
DECcou:INCads:INCbuf+1:LDAcou:BNEr1:LDA#
&50:JSRlatch:LDA#0:STAbits:RTS
 2100   .burn LDA#0:STAbuf:STAads:LDA#&3
0:STAbuf+1:LDA&414:STAcou:LDA#&50:JSRlat
ch:LDA#&54:JSRlatch:LDA#1:STApwr:LDA#&40
:JSRlatch:LDA#129:LDX#70:LDY#0:JSR&FFF4:
SEI:LDA#&CE:STA&FE6C:LDY#0
 2110   .bu0 JSRadl:LDA(buf),Y:STA&80:LD
X#0:STXbits:.b0 ASLA:BCSP%+4:INCbits:INX
:CPX#8:BNEb0:LDXbits:BEQbu3:.bu2 LDA#&50
:JSRlatch:LDA(buf),Y:STA&FE60:LDA#&10:.b
u1 BIT&FE6D:BEQbu1:NOP:NOP:DEX:BEQbu3:LD
A#&40:JSRlatch:JMPbu2
 2120   .bu3 LDA#&CE:STA&FE6C:LDA#&0:JSR
latch:NOP:LDA#0:STA&FE62:NOP:NOP:LDA&FE6
0:CMP(buf),Y:BEQago:STYbuf:STA&81:LDA#1:
STAbits:BNEex:.ago LDA#&42:JSRlatch:INY:
BNEbu0:DECcou:INCads:INCbuf+1:LDAcou:BNE
bu0:LDA#0
 2130   .ex STAbits:CLI:LDA#0:STApwr:LDA
#&50:JMPlatch
 2510   .adl LDX#0:LDAbuf+1:JSRdisp:TYA:
.disp PHA:LSRA:LSRA:LSRA:LSRA:JSRbkd:PLA
:AND#15:.bkd CMP#10:BCCbk1:CLC:ADC#7:.bk
1 CLC:ADC#48:STA&7C7C,X:INX:RTS
 2999   ]:NEXT:CALLinit:ENDPROC
```

into memory where it can be examined/saved/copied into another EPROM.

5) Blow the EPROM. More on this option below.

6) Examine memory. Produces a hex/ASCII dump of the contents of the buffer. Use the up/down cursor keys to move and <SPACE> to exit to menu.

7) Compare. This option compares the contents of the memory in the ZIF with what is in the buffer memory, byte by byte. This is a useful check that a freshly blown memory is a faithful copy of the original.

Back to option 5, the main task of this project. Blowing EPROMs has been made as easy as possible, select option 5 and you are away. Just one thing to bear in mind, make sure that the EPROM sitting in the ZIF socket is, if not virgin, at least fully erased. The programmer cannot program a EPROM that does not contain all &FFs. After each byte is programmed the computer will read it back and check it against what it thinks should be there. If there is a mis-match programming will terminate with an error message.

**Please note – the parts list shown last month contained numerous errors. Most are, dare we say, obvious and the values shown on the circuit diagram are correct.**



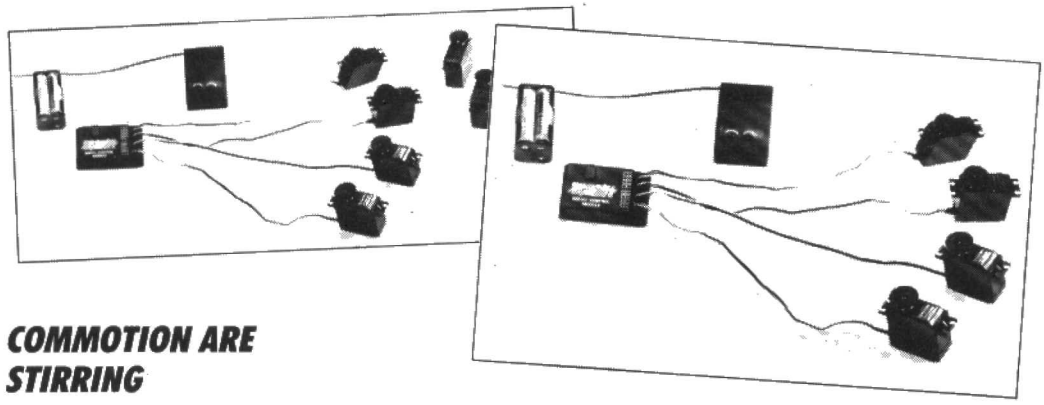
**Answers** – A 6502 – in some (sum) equals 13. B 5 (volts) the level of a logic 1.

FREE EVERY MONTH

# Your ROBOT

## BRITAIN'S FIRST ROBOTICS MAGAZINE

**Each month, Your Robot reports on the latest news from the world of low cost robotics. In addition in-depth features explain the theory behind the operation of robots and computer control systems.**



### COMMOTION ARE STIRRING

*Commotion, best known for their Beasty servo control system and the Snap EV1 electronic vision system, both for the BBC micro, are to launch a number of new products at the High Technology and Education Show. Commotion will be demonstrating an infra-red version of Beasty. This eight channel interface will be available for both the BBC and Commodore 64 micros and will be fully compatible with both the Beasty arm and mobile base. It will control up to eight servos within a minimum range of six metres.*

*Another new product is a version of the EV1 for the Commodore 64. This electronic vision system will allow the computer to store and display high resolution pictures and also to detect motion and recognise objects.*

*Further news from Commotion concerns the company's appointment as distributors for both the Fischer Technik and the Movit range of low cost robotic kits.*

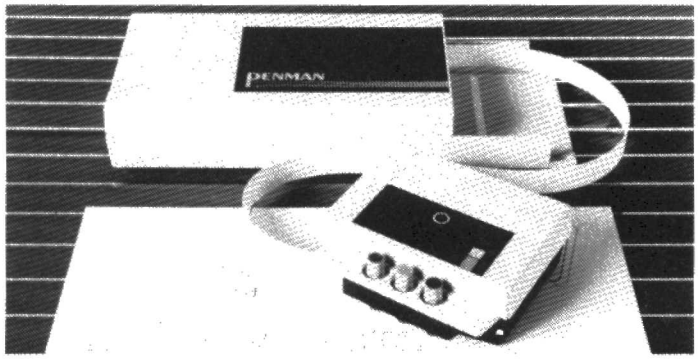
*Commotion will be found on stand 158 of the High Technology and Education Show which will be held at the Barbican between January 23rd, and 26th.*

### ROBOT BUILDERS CLUB

*Roboticists who wish to consult fellow enthusiasts will now be able to do so via their very own club. The Amateur Robotics User Club (ARUC) aims to act as a central network so that people may contact and consult with each other on their favourite subject.*

*Concerning itself solely with domestic and hobbyist robots, members will be kept informed by newsletter as to any research and developments which occur in the field. The association will also be concerned with sponsorship of design projects as well as with organising exhibitions and conferences. The first of these will be a conference in Cambridge in March 1985 and readers of* **Your Robot** *will doubtless be kept well informed of the project.*

*Membership may be corporate, affiliative or associate and more information may be obtained from the ARUC, 26 Mill Hill, Weston Colville, Cambridge, CB1 5NY.*




### PEN MIGHTIER THAN TURTLE

*The Penman robot plotter featured on Tomorrow's World some months ago is now in full production – but for the moment it won't work with Acorn's Bitstik CAD software.*

*The Penman is an innovative plotter, similar in design to a robot turtle. The only restriction on the Penman's movements is an umbilical cord which links it to the computer interface. The plotter prints in three colours, and manufacturers Penman Products claim a resolution of less than one thousandth of an inch.*

*Software is now available to drive the Penman from an IBM PC, Apple II, and BBC micro, but not in its most useful application, as a plotter for the Acorn/Robosystems Bitstik system. Acorn are responsible for developing the driver software but are not yet in a position to deliver. A company spokesman could not give a date when the software would be ready.*

*However in one month the Penman will be available to BBC owners as a £239 utility pack incorporating a mouse-driven menu, Basic program routines and LOGO drivers. It will also work with the MICAD 2D/3D CAD program which Ellis Hallwood are marketing from December at a price of £30.*

*The basic price of the Penman plotter is £217, which the company claim is half that of drum and flatbed plotters of a similar degree of accuracy. The device is expected to find a large market in education, business and design, and to capitalise upon this, work is in progress to develop programs for the Sinclair QL, RML 380Z, CBM 64, Atari, Apricot, Macintosh and TRS-80 machines.*

### PINT SIZED HERO

*Maplin, UK distributors for Heath Electronics, have launched a smaller version of their Hero robot. Appropriately introduced to the press at Harrods during the Christmas period, the robot is fully pre-programmed and has facilities for further personality expansion.*

*Hero junior is obviously designed with children in mind. He will sing songs, recite poetry, play games and as it comes complete with a built-in alarm system, can even be used as a rather crude security system. It'll even wake you up in the morning. Standing about a foot and a half tall, the unit is probably quite close to the layman's idea of what a personal robot should look like and, of course, to complete the idea, Hero jnr can even carry a tray of drinks.*

*The software cartridges that Maplin have made available suggest that they are also looking towards the educational market. However, the inclusion of packages of party songs and games could be seen as indicating that Hero jnr's true place may be as a rich man's toy. Nevertheless, a price tag of less than £600 for the kit form certainly makes Hero far more affordable than its big brother.*

*Further information can be obtained from Maplin Electronics, PO Box 3, Rayleigh, Essex. SS6 8LR. Telephone (0702 552911).*

**YOUR ROBOT NEWS DESK 01 251 6222**

**Your Robot,** 30/32 Farringdon Lane, London, EC1R 3AU. **Editor:** Gary Evans.

## LOW COST ROBOT BUILDING

# GETTING A GRIP

*Richard Sargent lends budding robot builders a hand in the form of some useful hints for those wishing to give their systems some grip on reality.*

There is something inherently fascinating about robot limbs. Perhaps it's the way in which the metal, motors, tubes and wires mimic the human counterpart. Or perhaps it's the funny movements and whirring sounds which aren't at all human that fascinate. I like the small red robot on commercial television which pops a well-known tea-bag into a cup for some hard-earned refreshment. It reminds me that there are really three types of robot: there is the *science fiction* model, on which the Japanese with their futuristic computer projects are doubtless beavering away, then there are the industrial robots like the tea-drinking model which are so primitive they're probably at the dinosaur level of robotic evolution, and lastly there are the so-called teaching robots. Teaching robots are models which exist to illustrate the principles of motion, leverage, torque, inertia and, of course, computer control to a generation of humans who will witness the arrival of the 5th generation computers – complete with their artificial intelligence and their ability to control industrial machinery.

A robot training kit may not seem to do an awful lot in relation to its cost. It certainly won't help around the house, but as a safe micro-driven version of the industrial machine which carries a five-figure price tag, it's got a lot going for it. Apart from the hands-on experience of driving an educational robot how else can a computer programmer move from the confines of the flat screen to the world of movement in three-dimensions?

**"... it is as well to consider the architecture of the arm ..."**

## TYPES OF ARM

Before tipping the contents of the LEGO box onto the floor, it would be as well to decide upon the design or *architecture* of the model arm. The architecture will determine the reach of the arm: all points in reach of the arm's gripper will be within a volume of air known as the *working envelope*. **Figure 1** shows four types of Robot Arm.

In theory, a gripper on the end of an arm may be taken to any point in a volume of space by reference to three axes, X, Y and Z. The minimum arm specification will therefore have three motors, one for each axis. But this does not take into account any dexterity that you might like to give to the gripper. Unless you intend to use an electromagnet or a hook for the gripping

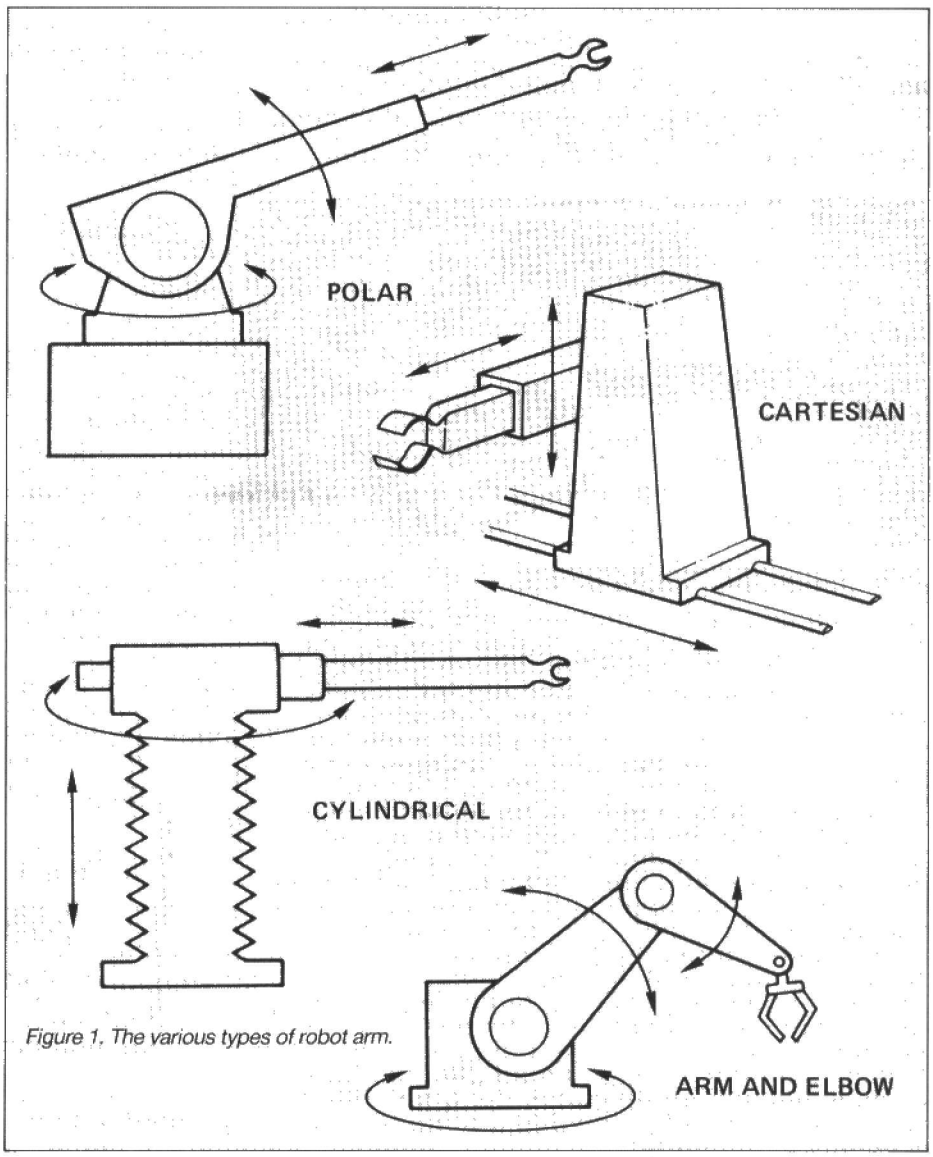



*Figure 1. The various types of robot arm.*

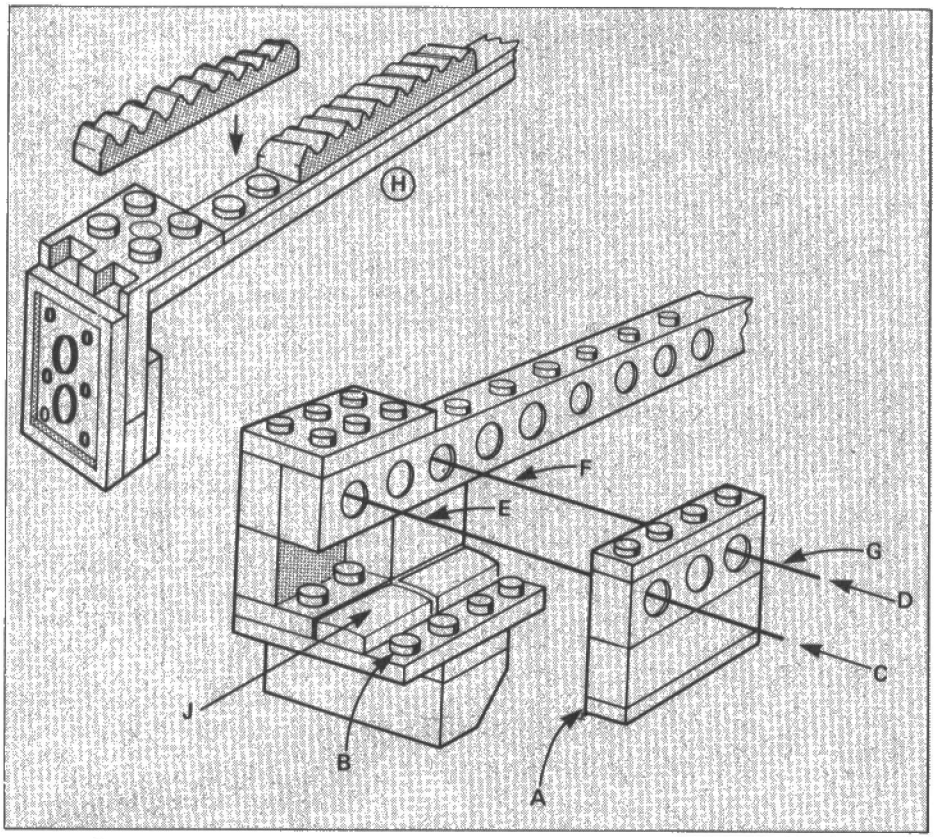



*Figure 2. Gripper constructed mostly of LEGO beams.*

device, you will certainly need a fourth motor to work the jaws/fingers of the grip. Dexterity is introduced by a "wrist" arrangement and the ability to roll, pitch and yaw will need three additional motors. It begins to become apparent why robot arms have at least six motors, and why they can never be particularly cheap. Unfortunately the "Low-Cost" subtitle of this series means that LEGO wrist movements are out of the question. To see what you're missing, try wiggling your hand!

When you swivel your wrist you are performing a "roll". Painting a wall in vertical strokes necessitates the movement of "pitch", while the action of removing a word with an eraser, palm on table, represents "yaw".

Even if the robot arm is limited to three axes and a gripper, there is still the difficult task of designing the structure. Looking at commercial devices gives some idea of the choices available. There are direct-drive models with a motor at each joint and models where the motors are all at the base but power-transmission is effected by cords, belts, rods or gears. A LEGO arm cannot really carry the weight of a motor directly on the upper arm joints so a design is needed where the motors and gearing are at the base, or possibly on the pivot of the lower arm, and where power-transmission is reasonably accurate and accomplished without bulky or heavy components.

## LOW COST OPTION

The best option for a LEGO robot arm is to use a design based on the polar variation of arm architecture. This is a three-motor design, with grip being either a simple hook or a small electromagnet, and as such will serve to illustrate the principles of computer control in a three-dimensional envelope. To further illustrate the technicalities of grip and wrist mechanisms separate stand-alone units can be built. Those of you with a taste for engineering (and a good supply of Technical LEGO) can set about marrying the complicated grippers to the arm structure if you so wish!

**"Many robot arms have at least six motors . . . therefore they can never be very cheap".**

## A LEGO ARM

The detailed description of the construction of a LEGO arm would take up quite a few thousand words and would require a dozen or so drawings and the Editor's budget doesn't stretch that far! Fortunately we can lean on the expertise of LEGO this month, since many of the models from their *Technical LEGO* series can be converted into respectable robot arms!

The polar-type arm is from the same family of machines as the traditional mobile-crane with extending jib. The LEGO model crane (Kit 855) fits the bill perfectly. The removal of its lorry-like features (cab and wheels) and the addition of three electric motors changes the scale and produces a low-cost polar robotic arm.

Alternatively, you might consider investing in the LEGO technical manual, item 8888, which contains instruction on how to build a variety of models, including a mechanical digger on caterpillar tracks. In this case, the removal of cab, bucket-scoop and tracks results in a quite reasonable arm-and-elbow robot! The advantage of this manual is that besides showing constructional details of some twelve specific models, it is also a useful source-book for engineering details which can be incorporated into any LEGO model.

The thorny problem of gripper-design is a nice engineering challenge to consider first.

## THE LEGO GRIPPER

Two useful parts kits to have are number 8710, which has a selection of beams, pulleys together with those most useful pieces, the *bevel gear* and *universal joint,* and number 877 which has small *swivel plates* and *rack-and-pinion* components. The mail-order pack 1228 has *differential gears* and rack, while pack 1227 has 36mm pulleys and two of the useful 40mm gear wheels.

Three versions of a LEGO gripper are presented. Version one **(Figure 2)** is made largely of LEGO beams and is a design based on the calliper instrument of the engineer. Block A fits onto plate B, but before fitting, two short axles C and D are inserted into the appropriate holes. Each axle carries a small gear-wheel (pinion-wheel) at E and F and they are effectively sandwiched when block A is fitted. Axle C freewheels, but axle D holds a pulley-wheel at G and can therefore receive power from a motor further down the arm. Unit H slides over flat-plates J and under pinion wheels E and F. The grip of this vice-like construction is powerful and effective.

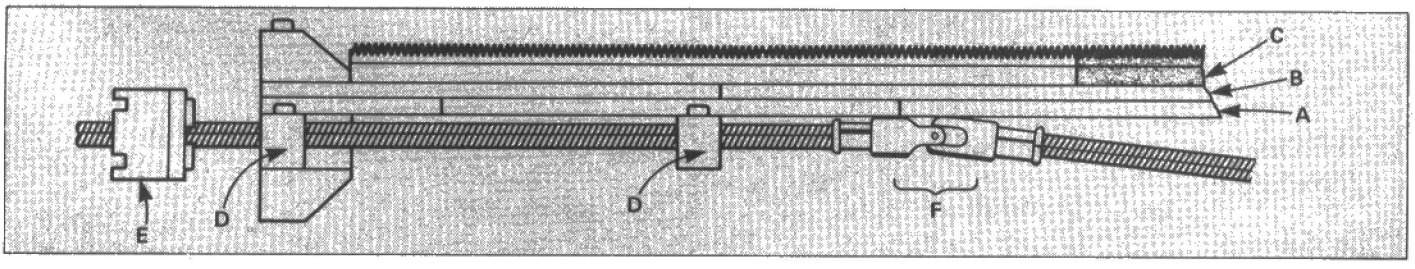



*Figure 3. Upper portion of an arm made from LEGO plates.*

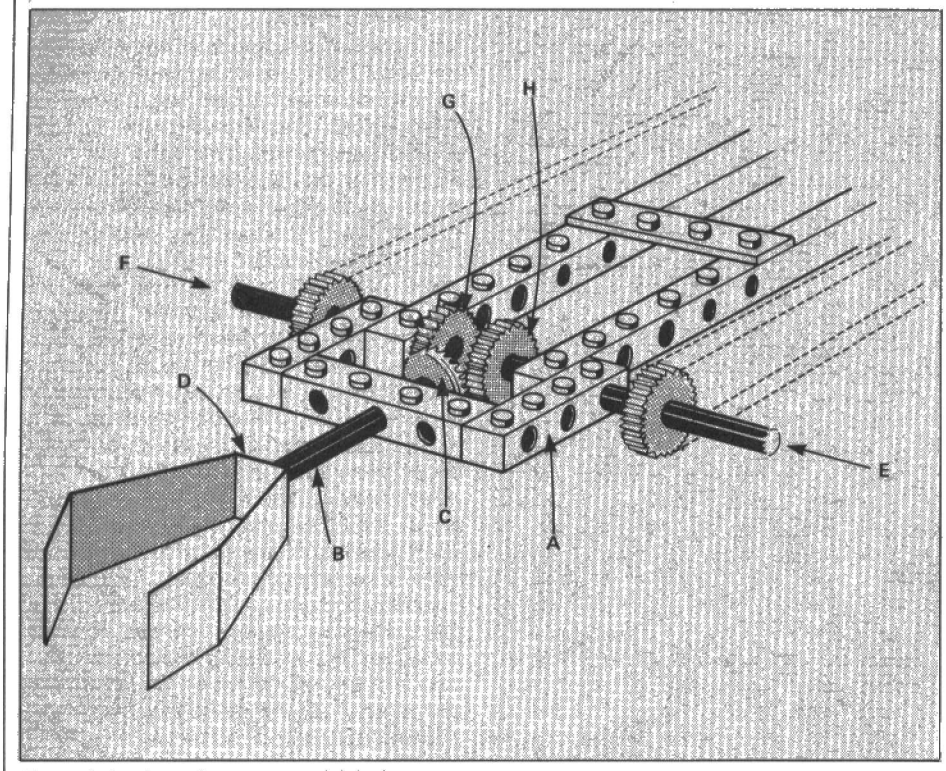


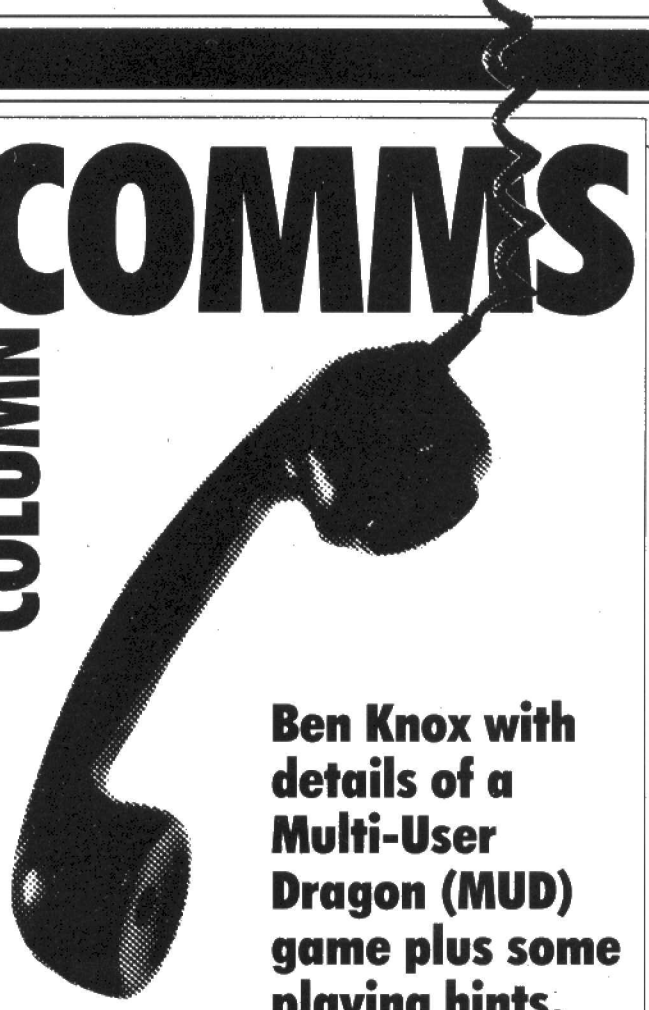

*Figure 4. Arm based on commercial design.*

Version two, **(Figure 3),** is the upper portion of an arm made from LEGO plates. The "16-stud" plates A and B are built up in layers and strengthened by rail-track pieces C. "3-hole" beams, D, project from the arm and carry the LEGO rod in the manner of a sliding piston. The gripper is a "round-block" and plate arrangement, shown end-on as E. Lower down the arm a rotating gear wheel drives the rod via universal joint F. This "piston-gripper" is a less powerful unit than the "caliper-gripper" since there are a number of links on the rod which are subject to disconnection, but it is perfectly adequate for the task of picking up small objects.

The final version is based on an actual design used on many commercial arms. The drawing of the LEGO in **Figure 4** has been simplified slightly so that the mechanical action can be visualised. Frame A carries axle B, onto which is firmly fixed bevel-gear C and grip-carriage D. The entire frame and grip-carriage assembly can pivot on axles E and F but only when bevel-gears G and H are rotating *in the same direction.*

When these two gears are rotating in opposite directions frame A is stationary and the motion is transmitted to the grip-carriage and a rotation of the "wrist" (axle B) occurs. So far, so good, but there is still the problem of achieving grip-movement. Professional wrists working on the bevel-gear principle have a drawstring threaded through axle B which can close the fingers of the grip against the pressure of a spring. The LEGO version will probably have to make do with a magnetic grip or hook, or possibly a solenoid-operated finger – unless you want to try and drill a hole through the length of axle B to take a drawstring. Any effort spent on this design is well worth it, however. The wrist action that can be performed, even in LEGO, can illustrate the Robot's ability to move test-tubes of water and to pour their contents without spilling any. **To be continued**

# COMMS COLUMN



## Ben Knox with details of a Multi-User Dragon (MUD) game plus some playing hints.

Adventure games have come a long way since the first one was written for a Dec-10 mainframe computer, many years ago. For adventures on microcomputers, cheap memory and improvements in data compaction techniques have enabled very complex games to evolve which include graphics, independant characters, and thousands of locations.

With all this sophistication, micro-based adventures still begin to pall after you become lost in a maze for the fiftieth time or when you realise that the so-called independent characters are only window-dressing with little, or no, ability to help.

Multi-User Dungeon (MUD) is the first multi-user adventure game. The original version resides on Essex University's Dec-10. Another version has been installed on Compunet, the network for Commodore 64 users. MUD was written by Roy Trubshaw in 1979, an undergraduate at Essex at that time. In 1980 Richard Bartle took the game over, expanding and tidying it up into its current form.

The scenario of the game is similar to many adventures: a mystical land with goblins and dwarfs in plentiful supply. There is an underground kingdom, a sea and an island. Players may become lost in forests, fall off of a cliff, go down a mine, drink poison, get tangled in a giant spiders web and so on.

On your travels around the land it is very likely that you will meet other people playing the game. You may talk to them, steal treasure which they may have collected, and even kill them.

When you access MUD for the first time, you will be asked for the name, sex and a password for your persona. Your persona is the character you will play the game as, so choose a name which you think will suit you! Few regular players use their real names, so don't be surprised if, once inside the game, you find people called Gandalf, Endora, Gorpli, Azaz, or anything else equally wierd.

New players always start at Novice level. There are a total of ten levels: Novice, Warrior, Champion, Hero, Super-Hero, Enchanter, Sorcerer, Necromancer, Legand and finally Wizard.

To increase your level you need to increase your points. Points are given whenever you collect treasure and deliver it to 'The Swamp', when you kill someone and sometimes you can be given them by a Wizard. If you are killed by another player, then you will have to start again from Novice level. If you are killed by a non-human, ie something generated by MUD, then your level will be retained for your next game.

As you go up through the levels you will become stronger, and you can try to cast spells on other people. Using spells, you can summon people to where you are, find out where treasure is, put someone to sleep, blind, silence, deafen or cripple them, and cure them. In fact anyone can attempt to cast a spell, the chance of success increasing with your level. Similarly, your victims chance of resisting your spell increases with his level. Wizards' spells always work.

The level of Wizard is a substitute for what is known as an end-game. In normal adventures, when you have got the maximum number of points you will usually be put into the final location which, up until now, you have been unable to find. This location, the end-game, may contain a treasure chest or something similar. After you have got to this place, that's it, you have finished.

In MUD, you are given an on-going end-game in becoming a Wizard. A Wizard can do almost anything: kill people, give them more points, move treasure, reset the whole game. Perhaps the most interesting command a Wizard has is Snoop. With Snoop, he can see what is on the screen of another player; watching all their mistakes and how they work out problems in the game.

Throughout the MUD game you are able to talk to the other players, either individually, or en masse. Through these communication facilities you are able to cant with others, perhaps to kill off another, stronger player who has been annoying you or to all help opening a portcullis.

To access MUD, you will need to use PSS. Essex's PSS address is: A2206411411. On the computer, type: LOG 2653,2653<CR>, the password is GUESS. There will be a short welcome message and a '.' will be displayed as the prompt. Type: RUN MUD, and you will be in the game. I usually play the game as 'JEFF'. See you there!

Below is a list of some commonly used commands:

## Guidelines for BBS use

1. Do not use false names or addresses.
2. Always go through the logging off procedure when you want to leave a BBS. Do not just hang up.
3. Check operation times of a BBS before calling.
4. If a voice answers, do not hang up. The BBS may be off-line for some reason, or you may have the wrong number.
5. If a BBS uses a 'ring-back' system, use the following procedure: dial the BBS number, allow it to ring once or twice, hang up, call back again – the computer should answer this time.

| Command | Abbreviation | Function |
|---|---|---|
| brief | — | Place descriptions not displayed |
| bye | — | leave game |
| drop <object> | DR | Drops object |
| flee <direction> | FL | Run away in specified direction |
| follow <name> | — | Follows other player |
| get <object> | G | Pick up object |
| go <direction> | — | Moves you in specified direction |
| help | H | Online help file |
| inventory | I | Lists objects which you are carrying |
| look | L | Displays description of current location |
| password | PW | To change your password |
| quickwho | QW | Displays names of other players |
| quit | Q | Leave game |
| "<message> | — | Sends message to other players in the same location |
| score | SC | Details of your current status |
| shout <message> | SH | Sends message to all other players |
| sleep | — | Sleep to regain energy |
| tell <name>, <message> | — | Tells only named player your message |
| who | — | Lists names and levels of other players |

# BCPL
# THE BARE FACTS

## Adam Denning describes two new QL language implementations and uses the opportunity to examine the philosophy behind their specification.

The QL is a Cambridge machine and BCPL is a Cambridge language. It was therefore only a matter of time before a compiler was produced for the QL. Again from Metacomco, the BCPL Development Kit is now available for £59.95. The package includes the compiler, a screen editor and a linker.

The manual describes the useage of all these programs, and also describes the language to those used to high level languages. It does not attempt to teach the language and refers the user to the only two good books on the subject: Martin Richards and Colin Whitby-Strevens *BCPL – The language and its compiler* and the Acornsoft *BCPL User Guide* written by Chris Jobson and John Richards.

### It's your choice

Whether you choose to program or not in BCPL depends on whether you like the language's philosophy. Like its daughter language C, BCPL is the only language which really allows you to get down to the nitty gritty of the machine. As an example of this the short QL BCPL listing **(Listing 1)** shown here simulates the CAT SuperBasic procedures shown last month which prints the filenames and lengths of all the files on a specified cartridge. This would be fairly difficult to do in languages like Pascal or BASIC.

### Executive action

The QL compiler is invoked by typing EXEC (or EXEC_W) mdv1_bcpl, prompting for the name of the source file, the code file, an error file and compiler options. The source file would of course have been prepared with the editor. The end product is a BCPL linker type file which needs to be linked in with the BCPL runtimes before it can be run. This is done with the **blink** program, which results in an EXECutable file (ie, a job) comprised of 68000 machine-code. This file is a lot larger than the file produced by the compiler, naturally enough, as it includes all the BCPL runtimes. Because of this, it is often more sensible to write programs in assembler if they approach the trivial.

The language follows the standard fairly well, with a bit of inspiration taken from the Acornsoft implementation. It also has floating point, using the same vector method adopted by Richards Computer Products for its BBC micro implementation.

There are a number of procedures to take advantage of the QL's facilities, but the present version does not allow you to change screen more or to alter the baud rate of the serial ports.

All microdrive files are automatically random access and there are various routines to take advantage of this. If you have special needs then the non-standard OPEN function can be used to access QDOS trap #3 routines directly, but it is recommended that you use the usual FINDINPUT and FINDOUTPUT functions.

Graphics and the real time clock are available, as are all the window facilities. Like most BCPL compilers it is very easy to include machine code procedures in programs via the global vector.

More interestingly, one can read and write file headers for microdrive files, enabling all sorts of file generation. The procedures SETGLOBALS and UNSETGLOBALS make it very easy to include overlays in large programs, such as assemblers and compilers.

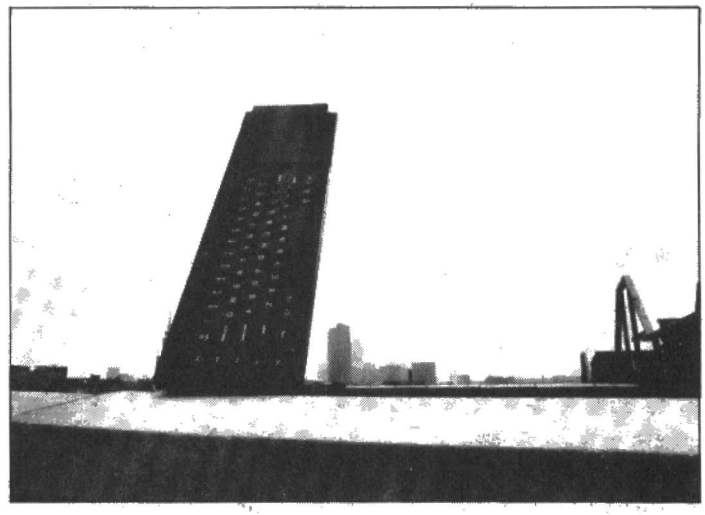
In fact that is where the beauty of BCPL lies – it is an ideal language for writing systems program,s having low level features coupled with the simplicity and clarity of a high level language. Most of Metacomco's products are written in BCPL, which shows the power of the language.

If BCPL is or could be your language then this compiler is highly recommended. It is easy to use and offers the capabilities of a wonderful language to all QL users frustrated by SuperBasic or unwilling to handle machine code.

**LISTING 1.**

```
GET   "libhdr"

MANIFEST $(
            open.dir = 4  //Open as directory key for OPEN
            no.buf = 0  //unbuffered file buffer length
            dir.header.length = 64  //Length of headers in directory
          $)

LET START() BE
$( LET drive,channel,more = ?,?,?
   LET device = "mdv1_"

   $( WRITES("*NCatalogue which drive? ")
      drive := READN() ; RDCH() //remove last character from buffer

      device%4 := drive ! '0'
      channel := OPEN(device,open.dir,no.buf)
      IF channel < 0 THEN STOP(channel)

      GET.DIR(channel)
      ENDREAD()
      SELECTINPUT(SYSIN)
      WRITES("*N*NCatalogue another drive?")
      more := RDCH() ; RDCH()  //Remove line feed
    $) REPEATWHILE CAPITALCH(more) = 'Y'
$)
/*
GET.DIR reads each of the file headers in the directory file into a
buffer and then processes and displays the information in each header as
appropriate
*/
AND GET.DIR(chan) BE
$( LET i,answer = ?,?
   LET header = VEC 16

   SELECTINPUT(chan)

   WHILE READBYTES(header,dir.header.length) NE 0 DO
   $(
      IF header!0 THEN  //this word is zero if the file has been deleted
      $(
         WRITES("*NFilename: ")
         answer := header%15  //This is the filename length
         FOR i := 1 TO answer DO
         WRCH(header%(15+i))  //These are the characters of the filename

         WRCH('=') ; WRITEN(header!0 - dir.header.length)

(defun sub2 (a) (difference a 2))

(defun fib (a) (cond
   ((eq a 1) 1)
   ((eq a 2) 1)
  (t (plus (fib(sub1 a)) (fib(sub2 a))))))

(defun fib-series (a)
  (messoff 3)
  (loop
   (until (eq a 31) (messon 255))
   (printc 'The blank 'Fib blank 'of blank a blank 'is blank (fib a))
   (setq a (add1 a) )))
```

# LISP



Languages are now coming thick and fast for the QL. The latest to join the list is QL Lisp from Metacomco. It costs £59.93 and is supplied with a manual describvving basic aspects of QL Lisp, such as invocation and use of the supplied screen editor. The manual will not help anyone who has never met Lisp before, and it refers you to the Acornsoft Lisp manual by Arthur Norman and gillian Cattell. The main reason for this is that QL Lisp is an extension of Acornsoft LISP and was written by the same person.

Apart from having all the Lisp specific functions (as opposed to BBC Micro specific functions like VDU) and predefined variables found in Acornsoft Lisp, a number of other routines have been added to take advantage of the QL and the 68008 processor. It was also an opportunity to add all those functions that people complained were missing in Acornsoft Lisp.

This Lisp now has turtle graphics, including the sophisticated QDOS capabilities such as circles and windowing. It has improved file handling and it is of course multi tasking. It was written in BCPL, which just goes to show how useful that language is!

Although a number of workers believe that micro-PROLOG has overtaken Lisp in its fundamental field of expert systems, the majority of Cambridge seems to disagree! It is rumoured that a QL micro-PROLOG written in C will eventially become available but until then many people will be satisfied by this Lisp. It is of course the language of brackets and supremely powerful list processing – from whence comes its name – and numerous other languages have stolen certain of its concepts.

One particularly interesting aspect of QL Lisp is that it is possible to create a Lisp program using the screen editor and then run it from the Lisp package by invoking one function. Its output can even be directed to another file if need be. It uses the integral QL real-time clock to time itself, so benchmark programs can easily be written.

Lisp is a language to which recursion has special significance. Although, as ever, there is no need to use this method of programming, Lisp programs are far easier to read if self-invoking or mutually invoking functions are written.

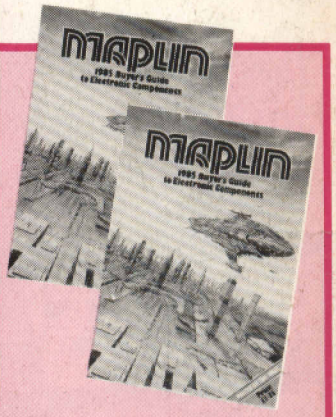
Lisp systems usually have nothing in them (no function definitions and no variables) until an image is loaded. The standard image supplied contains all the functions and identifiers so far described, as well as a host of others, so adding a program, which consist of function definitions, enlarges the object list which can subsequently be saved to microdrive or whatever. Whenever that image is reloaded Lisp is placed in exactly the same situation as it was when the image was saved.

If no other jobs are present when Lisp is loaded, the space available is over 50k, which is far more than the BBC Micro implementation (obviously!), so the package can be used for quite serious uses; it is not only an environment in which one can get used to the language.

Metacomco is eagerly awaiting the promised extra RAM for the QL, so that it can release its Cambridge Lisp, which is extremely large and one of the most comprehensive Lisps available. Meanwhile this QL Lisp is a very good implementation of the language and is an ideal opportunity to program in this most bracketed of languages!

# All new in the 1985 Catalogue


From a gentle purr to a mighty roar, the tightly controlled power of the beast is yours to command!



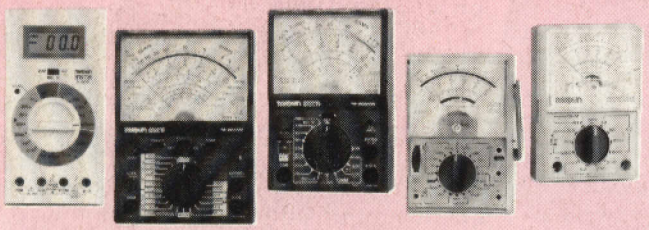

## PROFESSIONAL QUALITY HIGH POWER LOUDSPEAKERS

***A new range of superb quality loudspeakers.***
- ***★ Virtually indestructible high temperature voice-coil reinforced with glass-fibre***
- ***★ 100% heat overload tolerance***
- ***★ Advanced technology magnet system***
- ***★ Rigid cast alloy chassis***
- ***★ Linen or Plastiflex elastomer surrounds***
- ***★ 5-year guarantee (in addition to statutory rights)***

Available in 5, 8, 10, 12, 15 and 18 inch models with 8Ω and some 16Ω impedances and with input powers ranging from 50W to 300W e.g.

5in. 50W 95dB 8Ω: XG39N / 16Ω: XG40T £17.95§
8in. 100W 98dB 8Ω: XG43W £29.95§
10in. 100W 100dB 8Ω: XG46A £29.95§
12in. 100W 101dB 8Ω: XG49D £29.95§
12in. Twin Cone 100W 100dB 8Ω: XG50E / 16Ω: XG51F £31.95§
*Note - the output power doubles for each 3dB increase (ref 1W @ 1m).*

## PRECISION GOLD MULTIMETERS



***A new range of very high quality multimeters offering truly amazing quality at the price.***

Pocket Multimeter, 16 ranges, 2000Ω/V DC/AC £6.95§ (YJ06G)
M-102BZ with Continuity buzzer, battery tester and 10A DC range, 23 ranges, 20,000Ω/V DC £14.95§ (YJ07H)
M-2020S with Transistor, Diode & LED tester and 10A DC range, 27 ranges 20,000Ω/V DC £19.95§ (YJ08J)
M-5050E Electronic Multimeter with very high impedance, FET input, 53 ranges including peak-to-peak AC, centre-zero and 12A AC/DC ranges £34.95§ (YJ09K)
M-5010 Digital Multimeter with 31 ranges including 20Ω and 20μA DC/AC FSD ranges, continuity buzzer, diode test, and gold-plated PCB for long-term reliability and consistent high accuracy (0.25% +1 digit DCV) £42.50§ (YJ10L)

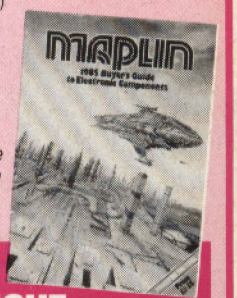
***Our huge range of top quality electronic components at very competitive prices are all detailed in our catalogue, and with well over 600 new lines in our 1985 edition and many design improvements, it's well worth getting a copy. Here are just a few examples from the catalogue. (The items below are NOT kits).***

★ Most phono and jack plugs now with integral strain relief sleeve - gold-plated types also available from 14p (gold from 70p)
★ Stereo Disco Mixer with cross-fade, talk-over, cue monitoring, aux input, slide controls. Only £58.95 (AF99H)

★ 10-Channel Stereo Graphic Equalisers - 3 models - basic; with peak level meter; and with spectrum analyser - from £77.95

★ Digital Delay Line permits Slap-back, Doubling, Flanging, Chorus and Echo. 11 controls. Only £195.00 (AF98G)
★ Video Enhancer improves picture quality when recording from one VTR to another, and with TV's with monitor input. Only 28.95 (XG59P)
★ Detailed descriptions of the exciting new 74HC range of IC's which combine the advantages of CMOS and TTL. From 46p
★ Keyboards: sloping keys, two-tone grey, mounted in steel frame, very smart cases (extra) available. 61 keys, only £33.95 (YJ12N)
79 keys, only £37.95 (YJ13P)
★ 1% Resistors now 50ppm/°C, 0.4W, only 2p each!
★ Auto transformers 120/240V 50VA, £10.75§ (YJ56L). 100VA £14.95§ (YJ57M). 150VA £16.95§ (YJ58N). 250VA £21.95§ (YJ59P).
★ Digital Clinical Thermometer. Only £13.95 (FK51F)

*Check our 1985 Catalogue for all our other fascinating new lines.*




## 1985 CATALOGUE

Pick up a copy now at any branch of W.H. Smith or in one of our shops. The price is still just £1.35, or £1.75 by post from our Rayleigh address (quote CA02C).

AMERICAN EXPRESS · map · Access · VISA

***☎ Phone before 2pm for same day despatch.***

*N.B. All our prices include VAT and Carriage. A 50p handling charge must be added if your total order is less than £5 on mail order (except catalogue).*

**MAPLIN ELECTRONIC SUPPLIES LTD.**

Mail Order: P.O. Box 3, Rayleigh, Essex SS6 8LR. Tel: Southend (0702) 552911
SHOPS
- BIRMINGHAM Lynton Square, Perry Barr, Tel: 021-356 7292.
- LONDON 159-161 King Street, Hammersmith, W6. Tel: 01-748 0926.
- MANCHESTER 8 Oxford Road, Tel: 061-236 0281.
- SOUTHAMPTON 46-48 Bevois Valley Road, Tel: 0703 25831.
- SOUTHEND 282-284 London Rd, Westcliff-on-Sea, Essex. Tel: 0702-554000

*Shops closed all day Monday.*

Post this coupon now for your copy of the 1985 catalogue.
Price £1.35 + 40p post and packing. If you live outside the U.K. send £2.40 or 11 International Reply Coupons. I enclose £1.75.

Name .....
Address .....
.....
.....

E&CM 2/85

***§ Indicates that a lower price is available in our shops.*** ***All offers subject to availability.*** ***Prices firm until Feb 9th 1985.***